View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Waterloo's Institutional Repository

# A Collaborative Architecture for Distributed Intrusion Detection System based on Lightweight Modules

by

Safaa Zaman

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2009

© Safaa Zaman 2009

# AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Safaa Zaman

# Abstract

A variety of intrusion prevention techniques, such as user authentication (e.g.: using passwords), avoidance of programming errors, and information protection, have been used to protect computer systems. However, intrusion prevention alone is not sufficient to protect our systems, as those systems become ever more complex with the rapid growth and expansion of Internet technology and local network systems. Moreover, programming errors, firewall configuration errors, and ambiguous or undefined security policies add to the system's complexity. An Intrusion Detection System (IDS) is therefore needed as another layer to protect computer systems. The IDS is one of the most important techniques of information dynamic security technology. It is defined as a process of monitoring the events occurring in a computer system or network and analyzing them to differentiate between normal activities of the system and behaviours that can be classified as suspicious or intrusive.

Current Intrusion Detection Systems have several known shortcomings, such as: low accuracy (registering high False Positives and False Negatives); low real-time performance (processing a large amount of traffic in real time); limited scalability (storing a large number of user profiles and attack signatures); an inability to detect new attacks (recognizing new attacks when they are launched for the first time); and weak system-reactive capabilities (efficiency of response). This makes the area of IDS an attractive research field. In recent years, researchers have investigated techniques such as artificial intelligence, autonomous agents, and distributed systems for detecting intrusion in network environments. This thesis presents a novel IDS distributed architecture – Collaborative Distributed Intrusion Detection System (C-dIDS), based on lightweight IDS modules – that integrates two main concepts in order to improve IDS performance and the scalability: lightweight IDS and collaborative architecture.

To accomplish the first concept, lightweight IDS, we apply two different approaches: a features selection approach and an IDS classification scheme. In the first approach, each detector (IDS module) uses smaller amounts of data in the detection process by applying a novel features selection approach called the Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF). This approach improves the system scalability in terms of reducing the number of needed features without degrading the overall system performance. The second approach uses a new IDS classification scheme. The proposed IDS classification scheme employs multiple specialized detectors in each layer

of the TCP/IP network model. This helps collecting efficient and useful information for dIDS, increasing the system's ability to detect different attack types and reducing the system's scalability.

The second concept uses a novel architecture for dIDS called Collaborative Distributed Intrusion Detection System (C-dIDS) to integrate these different specialized detectors (IDS modules) that are distributed on different points in the network. This architecture is a single-level hierarchy dIDS with a non-central analyzer. To make the detection decision for a specific IDS module in the system, this module must collaborate with the previous IDS module (host) in the lower level of the hierarchy only. Collaborating with other IDS modules improves the overall system accuracy without creating a heavy system overload. Also, this architecture avoids both single point of failure and scalability bottleneck problems.

Integration of the two main concepts, lightweight IDS and a distributed collaborative architecture, has shown very good results and has addressed many IDS limitations.

# Acknowledgements

For a long time, I have dreamt of this moment. The moment of realizing my dream: finishing my PhD degree and returning home to Kuwait. At the same time I will feel sad for leaving behind the people who supported me, gave me a hand when I needed, shared my hard times and always helped me to find my path. To all of these people, I am grateful.

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Fakhri Karray, for the opportunity of working with him and for his support and gentle guidance during my research. He led me through many fruitful discussions and has been a constant source of motivation, guidance, encouragement, and trust. His invaluable suggestions and ideas have helped me walk through each stage of my research, while his passion and extraordinary dedication have inspired me to work harder and succeed.

My appreciation is also extended to my committee: Prof. Sagar Naik, Prof. Liang-Liang Xie and Prof. Kumaraswamy Ponnambalam for offering many insightful comments, which improved my work; and to the external examiner Dr. Azzedine Boukerche who kindly agreed to assess my work.

Throughout my research, I was fortunate to meet and benefit from the support of many individuals at the Pattern Analysis and Machine Intelligence (PAMI) lab. My warm thanks go to my fellow PAMI members, in particular Dr. Mohammed El-Abd, Dr. Richard Khoury, and Arash Abghari for their generous help with technical issues, and for their valuable discussions and feedback. I am thankful to the PAMI administrative staff for all their cheerful assistance during my time in the program. In particular, I wish to thank Wendy Boles, Lisa Hendel, Heidi Campbell, Rosalind Klein, Sue Havitz, and Karen Critchley for providing me with a convenient research environment.

Many thanks also go out to Kuwait University for providing the financial support that made this work possible. I am also very grateful to Prof. Hassan Merza, Dr.Fatima Nazar, and my academic advisors Nikki Chy and Stacy Dellinger for their help and support.

I am deeply indebted to Sheikh Sabah Al-Nasser Al-Saud Al-Sabah for his amazing support and encouragement of my research work. My special thanks go to my friend Dr. Mohamed Omran for his valuable help and support. I also owe a great magnitude of thanks to Kuwait Society Engineers (KSE), in particular Eng. Talal Al-Qahtani, Eng. Saud Al-Oteibi, Saleh Al-Moteri, Eng. Hummod Al-Zeabe, Eng. Meajeb Al-ajmi, and Eng. Ahmed Al-Doseri for helping me overcome some stressful situations, and being such great brothers.

During this journey, I tragically lost my beloved mother, who had always dreamt of this day. I would like to express my deep gratitude and love to her for everything that she offered to me without any demand on her part. Her endless love, honest prayers, and true dedication to my cause made this achievement possible. Also, my warmest thanks go to the memory of my father. He is and remains always a great role model for me; though I cannot see him, I always feel that he is with me. Their spirits will be with me forever. I am deeply in debt to my beloved sister Hanaa Zaman for her unconditional support, love, and persistent confidence in me, which has lightened the load on my shoulders.

Finally, I would like to say “Thanks a lot!” to all of you: without you I would not have been able to finish this work.

*For my Beloved Mother*

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| | |
|---|---|
| AAFID | Autonomous Agents for Intrusion Detection |
| ACO | Ant Colony Optimization |
| AIDS | Application-layer Intrusion Detection System |
| ANN | Artificial Neural Network |
| AppIDS | Application-based Intrusion Detection System |
| ARP | Address Resolution Protocol |
| ASIM | Automated Security Incident Measurement |
| BE | Backward Elimination |
| BPL | Back Propagation Learning |
| C-dIDS | Collaborative Distributed Intrusion Detection System |
| CPN | Colored Petri Net |
| CR | Classification Rate |
| CSLIP | Compressed Serial Line Internet Protocol |
| CSM | Cooperative Security Managers |
| DARPA | Defense Advanced Research Projects Agency |
| DDoS/DDOS | Distributed Denial of Service |
| DHCP | Dynamic Host Configuration Protocol |
| dIDS/DIDS | Distributed Intrusion Detection System |
| DNS | Domain Name System |
| DoS /DOS | Denial of Service |
| DR | Detection Rate |
| EMERALD | Event Monitoring Enabling Response and to Anomalous Live Disturbances |
| ESVDF | Enhanced Support Vector Decision Function |
| FIRE | Fuzzy Intrusion Recognition Engine |
| FL | Fuzzy Logic |
| FN | False Negative |
| FNT | Flexible Neural Tree |
| FP | False Positive |
| FPR | False Positive Rate |
| FS | Forward Selection |
| FTP | File Transfer Protocol |

| | |
|---|---|
| GA | Genetic Algorithm |
| GrIDS | Graph-based Intrusion Detection System |
| HIDS | Host-based Intrusion Detection System |
| HTTP | Hyper Text Transfer Protocol |
| ICMP | Internet Control Massage Protocol |
| ICN | Integrated Computer Network |
| IDES | Intrusion Detection Expert System |
| IRS | Intrusion Recovery System |
| IDRS | Intrusion Detection and Response System |
| IDS | Intrusion Detection System |
| IP | Internet Protocol |
| IPS | Intrusion Prevention System |
| IRCP | Internet Relay Chat Protocol |
| LAN | Local Area Network |
| LDA | Linear Discriminant Analysis |
| LIDS | Link-layer Intrusion Detection System |
| MAC | Media Access Control |
| MAIDS | Mobile Agent Intrusion Detection System |
| MLP | Multi Layer Perceptron |
| NADIR | Network Anomaly Detector and Intrusion Reporter |
| NC-dIDS | Collaborative Distributed Intrusion Detection System with non-specialized IDSs |
| NIDS | Network-based Intrusion Detection System |
| NN | Neural Network |
| PCA | Principle Component Analysis |
| PHIDS | Parallel Hierarchical Intrusion Detection System |
| POP/POP3 | Post Office Protocol |
| PPP | Point to Point Protocol |
| PSO | Particle Swarm Optimization |
| R2L | Unauthorized access from a Remote machine |
| RARP | Reverse Address Resolution Protocol |
| RAS | Remote Access Server |
| RBF | Radial Basis Function |

| | |
|---|---|
| RIDS | Router-based Intrusion Detection System |
| RIP | Routing Information Protocol |
| SC | Soft Computing |
| SC-dIDS | Collaborative Distributed Intrusion Detection System with specialized IDSs |
| SFT | Software Fault Tree |
| SHIDS | Serial Hierarchical Intrusion Detection System |
| SLIP | Serial Line Internet Protocol |
| STAT | Transition Analysis Technique |
| SVDF | Support Vector Decision Function |
| SVM | Support Vector Machine |
| TCP | Transmission Control Protocol |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| TIDS | Transport-layer Intrusion Detection System |
| TN | True Negative |
| TP | True Positive |
| U2R | Unauthorized access to Root |
| U2Su | Unauthorized access to Super user |
| UCI | Irvine Machine Learning Repository |
| UDP | User Datagram Protocol |
| VLAN | Virtual Local Area Network |
| WAN | Wide Area Network |
| WLAN | Wireless Local Area Network |

# Chapter 1
# Introduction

*Information security* plays an important role in all aspects of life, in particular the protection of an organization's valuable resources, such as information, hardware, and software. Therefore, information security is defined as a process of protecting data from unauthorized access, use, disclosure, destruction, modification, or disruption. It is concerned with ensuring that information-related risks are assessed, appropriate controls are implemented to manage those risks, and that the adequacy of those controls is monitored on a regular basis. Generally, discussion of information security falls under three generic headings:

1. **Confidentiality:** This is a requisite for maintaining the privacy of people whose personal information the organization holds.
2. **Integrity:** This means that data cannot be created, changed, or deleted without authorization. It also means that data stored in one part of a database system is in agreement with other related data stored in another part of the database system (or on another system).
3. **Availability:** This means that the information, the computing systems used to process the information, and the security controls used to protect the information are all available and functioning correctly when the information is needed.

The field of information security has evolved rapidly in recent years because of the swift growth and widespread use of electronic data processing, and also of business conducted through the Internet and other computer networks (LAN, WAN, etc.). These application areas make networks an attractive target for abuse and thus an area of vulnerability. At the same time, the tools of the intruder and the hacker have improved substantially. In order to both combat the growing number of attacks and to maintain critical information services, both academic and industry groups have been developing systems to monitor networks and to raise alarms over suspicious activities. These systems are called Intrusion Detection Systems (IDS).

*Intrusion Detection* is defined as “the problem of identifying individuals who are using a computer system without authorization (i.e., crackers) and those who have legitimate access to the system but are abusing their privileges (i.e., insider attack: threat)” [124]. An *Intrusion Detection System (IDS)* gathers and analyzes information from various areas within a computer or a network to identify

possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization). An IDS is designed to detect unscrupulous activities that compromise the confidentiality, integrity, or availability of network or computer systems and to analyze what happens – or what has happened – to indicate that the computer has been misused. The IDS does not eliminate the use of a preventive mechanism, but rather works as a second defense mechanism behind a firewall, which can monitor the network while not affecting network performance. In conclusion, an IDS is the whole process that detects, audits, tracks, and identifies unauthorized access and abnormal phenomena actions or events in the system. It can identify whether the system is being accessed as it happens and take the appropriate actions to cut off network connections, record events, and raise an alarm. It can also remind the system administrators to take proper measures. More details on IDS are given in the next chapter.

Recently, a number of innovative approaches and new models for IDS have been proposed. But while many of the proposed techniques have relatively improved some of the shortcomings of the earlier approaches, still a number of issues remain: low detection accuracy, low real-time performance, and limited scalability. These problems make the area of IDS an attractive and open research field. In recent years, researchers have investigated a variety of different computational tools to improve IDS performance and overcome some of its limitations, such as Soft Computing (SC) techniques [8], [16], [19], distributed systems [41], [61], [98], and autonomous agents (AA)[44], [121], [99]. Still, a lot more needs to be done to deal with new technologies and tools developed by intruders to break the systems.

In this thesis, we try to overcome some of IDS limitations by proposing a new dIDS architecture through the integration of two main concepts. The first concept is accomplished by using a lightweight IDS module. Each IDS module used allows the detection process to function with a smaller dataset. To build a lightweight IDS module, we apply two different approaches: features selection, and an IDS classification scheme. The first approach is accomplished by using SC tools. We use a novel features selection algorithm called Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF). The Fuzzy ESVDF is an iterative algorithm based on a Support Vector Decision Function (SVDF) and Forward Selection (FS) approach. A fuzzy inferencing model is used to select the appropriate features set, in order to improve the performance of the IDS in terms of accuracy and efficiency (training time and testing time) [118], [119]. For the second approach, the IDS classification scheme [152] categorizes the IDS into four types depending on the TCP/IP network model: Application layer IDS (AIDS); Transport layer IDS (TIDS); Network layer IDS (NIDS); and

Link layer IDS (LIDS). Each of these IDS types is dedicated to a specific network device, so the detection process will be distributed among all TCP/IP network model layers though the network devices. Chapter 3 will provide additional detail about this proposed system.

For the second concept, the lightweight IDS modules will be integrated using a distributed collaborative architecture called Collaborative Architecture for dIDS (C-dIDS). This architecture, C-dIDS, contains a single-level hierarchy collaborative dIDS. To make the detection decision for a specific IDS module in the system, this IDS needs to collaborate with the previous IDS in the lower level only. The transferred data can then be dispatched between the detectors with only crucial data (just one bit of information). More details about this architecture will be provided in Chapter 4.

This chapter starts with an overview of IDS in Section 1.1. Then, the motivations and goals behind this thesis are discussed in Section 1.2. In Section 1.3, we present the thesis organization and describe the content of each chapter.

## 1.1 Problem Statement

The field of information security has grown and evolved substantially in recent years because of the rapid growth and widespread use of electronic data processing, and of business conducted through the Internet and other computer networks (LANs, WANs, etc). These application areas make networks attractive targets for abuse. At the same time, the tools of the intruder and the hacker have improved substantially. Facing these daunting challenges, industry and academic institutions are working hard to develop new devices, new approaches, and new security mechanisms to counter the challenges from malicious intruders. These efforts have resulted in a great variety of security products such as firewalls, encryption, authentication, vulnerability checking, and other measures. Nevertheless, most computer systems are still susceptible to attacks from hackers, so it is essential to establish a second line of defense for these systems in the form of an Intrusion Detection System (IDS).

IDS [50], [63], [122] play an important role in achieving the survivability of information systems and ensuring their safety from attacks. They aim to protect the availability, confidentiality, and integrity of critical network information systems by analyzing what happens or has happened during an intrusion, and attempting to identify signs that a computer has been misused. They can also take appropriate actions to sever network connections, record events, raise alarms, and remind system administrators to take proper measures.

IDS are usually classified as host-based or network-based. Host-based systems [32], [123], base their decisions on information obtained from a single host (usually log files, network traffic to and

from the host, or information on processes running on the host), while network-based systems [45], [56] obtain data by monitoring network traffic between hosts, and are usually run on a separate machine.

Most current IDS technology still suffers from three main problems which limit their detection ability: low detection accuracy (registering high False Positive alarms and False Negative); low real-time performance (processing large amounts of traffic data in real time); and limited scalability (storing a large number of user profiles and attack signatures).

Our proposed approach overcomes these limitations by integrating two main concepts: (1) Using lightweight IDS modules and (2) Having a novel distributed collaborative architecture for the IDS. Another key effort in our approach is that directed towards improving system robustness, extensibility, configurability, and security.

## 1.2 Thesis Motivation and Contributions

The ideal approach for computer security is to establish and implement a security policy that prevents any intrusion through the use of security measures. However, traditional preventive measures are not always sufficient, for the following reasons:

- Bug-free software is seldom attainable.
- It is difficult to change user and organization behaviour, to oblige all users to follow diligently security policy.
- Human errors in operations and maintenance are unavoidable; these errors can cause serious security loopholes.
- The security measures and controls themselves can be compromised: for instance, the cryptographic algorithms can be cracked, given sufficient time and computing power.
- It is almost impossible to prevent insider attacks because inside users naturally have greater access to the system than do outside attackers.
- The cost of setting up a totally secure system is very high, which discourages their implementation.

Because of the above difficulties, we need to use other alternative or complementary techniques to protect and secure our systems. One of the major techniques is the Intrusion Detection System (IDS). Intrusion Detection is another type of security tool that must be created to protect and secure the

information resources in the system. It complements firewalls by allowing a higher level of analysis of traffic on a network, and by monitoring the behaviour of the sessions on the servers. In addition, it possesses such special characteristics and benefits as:

- Networks are complex and difficult to monitor: an IDS can help reveal potential network security problems by documenting network status.
- An IDS highlights intrusion traces, which help to identify and eliminate the security flaws that enabled these intrusions in the first place.
- An IDS can assess the integrity of critical system and data files.
- An IDS provides real-time reporting of break-ins, allowing the system administrator to take immediate action, lessening potential damage.
- In contrast to a firewall, an IDS is a passive system that does not influence network traffic. Thus, most people attacking or trying to circumvent a system will not recognize the intrusion detection node. In addition, an authorized user can log on without interruption.

The current state of IDS technology is not yet fully reliable, which makes the area of IDS an attractive and still open research field. A major problem with current IDS is their inability to guarantee intrusion detection (**low accuracy**): the current IDS technology is not accurate enough to provide reliable detection. This problem will lead to a high rate of false alarms (False Positives), and missed alarms (False Negatives). A common complaint is that the large number of False Positives and Negatives generated by Intrusion Detection Systems makes it hard to filter out false attacks without potentially missing genuine attacks. Moreover, this low accuracy can lead to an incident handling problem: that is, security administrators are uncertain how to respond to mitigate the risks if a certain degree of accuracy cannot be achieved. There is no decision rule associated with each alert to tell the security administrator whether he should ignore the alert or simply terminate the suspicious session.

Another major problem is the speed of detection (**low efficiency**). The size of the feature space is obviously very large, which leads to slow training and testing processes, heavy computational resources, and low detection accuracy. Moreover, computer networks have a dynamic nature in the sense that the data within them are continuously changing. Therefore, in order to detect an intrusion accurately and promptly, the system has to operate in real time.

In addition to the problems outlined above, there are other limitations, such as:

**Inability to detect new attacks:** The ability to recognize new attacks when they are launched for the first time is very low; this reduces the overall system performance.

**Limited scalability:** The IDS is unable to achieve reliable scalability to gather and analyze the high-volume of audit data correctly from the distributed host, which may cause severe network performance degradation.

**Lack of extensibility:** It is difficult to extend the scope of IDS or reconfigure/add capabilities to the IDS.

**Difficult configurability:** The IDS is unable to configure itself easily to the local requirements of each host or each network component.

**Monotonic analysis:** Many network intrusions exploit the multiple points of a network. Thus, from a single host, they might appear to be just a normal mistake. But if they are collectively monitored from multiple points, they can be clearly identified as a single attack attempt.

**Low robustness:** In many cases, the IDS itself may fall under attack from a threat seeking to disable it. An IDS should itself be resistant to attacks, should exhibit a high degree of fault tolerance, and allow for graceful degradation.

**Low reliability (Point of Failure):** For most single IDS, if an intruder can somehow prevent the IDS from working, the whole network is without protection.

Recently, a number of innovative approaches and new models for IDS have been proposed to improve IDS efficiency and performance, such as Distributed IDS (dIDS). The dIDS [89], [77] is one of several options that allow computation load and diagnostic responsibilities to be distributed throughout the network. It performs distributed data collection (and some pre-processing) by using modules distributed in different hosts, which monitor separately and communicate and cooperate with each other. The dIDS can provide the foundation for a complete solution to the complexities of real-time detection, while maintaining fault-resistant behaviour. It has scalability to detect general attacks or a specific attack. In addition, each module can be added to or removed from the system without altering other system components, because they operate independently. Also, the system's modules can be configured or upgraded without disturbing the rest of the system, as long as their external interface remains the same.

Another approach used to improve IDS efficiency is Soft Computing (SC). In general, applications of SC are widely used with IDS, either for a detection model or for the generation of intrusion features selection. They are suitable for handling such subjective estimates for a number of reasons:

- fast recognition and classification;
- learning abilities;
- adaptability;
- flexibility;
- low solution cost;
- fast computing;
- ease of design;
- ability to generalize from learned data;
- not easily misled by small variations in intrusion patterns;
- modular with both misuse and anomaly detection components.

Researchers have proposed several approaches in this regard. Some researchers are more interested in SC techniques for such detection models as Fuzzy Logic (FL) ) [55], [56], Genetic Algorithms (GA) [ 57], [14], [43], Neural Networks (NN) [125], [16], [8], Probabilistic Techniques [126], AdaBoost [127], Immune System [128], and SVM [10], [17], [18]. Still others are interested in SC techniques for IDS features selection models such as NN [63, 64], GA [65, 66], SVM [63, 69], and other optimization tools [71, 72].

Despite advances in research on intrusion detection technologies, the current IDS technology is not accurate enough to provide reliable detection. Therefore, the main emphasis of this thesis is to improve IDS accuracy, time performance, and scalability by combining two main concepts: lightweight IDS, and a distributed collaborative architecture.

(1) Lightweight IDS

The first concept is being lightweight. To build a lightweight IDS module, we need to reduce the amount of data/features needed to achieve successful detection by applying two different approaches. The first approach is to use dimensionality reduction techniques (features selection approach). The second approach is to use an IDS classification scheme. By using lightweight IDS, it will satisfy the following requirements:

**Efficiency**: A lightweight IDS can improve the generalization performance of intrusion detection and make the detection more time-efficient. Faster training and testing helps to build an efficient IDS and provides ease of maintenance or modification of the IDS. Furthermore, a small number of input features leads to a reduction in execution times, which is important for the on-line detection of attacks.

**Accuracy:** By applying the proposed lightweight approach, which contains a new classification scheme, the overall system detection ability will be improved in three ways. First, each IDS type can be specialized to detect a specific category of attacks, depending on the layer. For example, to place an IDS in the router, we need to use NIDS, which has extensive information on router attacks behaviours. Secondly, by distributing the IDS through the TCP/IP layers as the second level defense after the firewall; the firewall will be supported by IDS and overall system security will be improved. Furthermore, it is known that one of the major issues in network security is securing network devices, which are represented as system entry points for the attacks. Hence, by designing a specialized IDS for each one of them, overall system performance will be improved.

**Scalability**: A lightweight IDS can improve scalability by reducing the amount of network load on each IDS module in the system. Thus, the system becomes scalable enough to be able to work correctly and efficiently with increased traffic on the network.

**Generality:** By splitting the detection process into different layers (levels) in the network according to the proposed classification scheme, each IDS module will be specialized to detect a specific attack type, which increases its ability to capture all or almost all known attacks.

**Intrusion's Influences Reduction:** Detection attacks in the first stages (higher or lower TCP/IP layer) before they enter the network, will reduce any damage that may occur.

**Extensibility:** By using a lightweight and specialized IDS, system extensibility will be improved. To extend the system, we need only add an IDS to the network device that we used to extend it. This architecture allows for computation to be performed at any point where enough information is available.

**Flexibility:** Because the lightweight IDS can be easily deployed on almost any node of a network with minimal disruption to operations, they can be added and removed from the system without altering other components.

**Configurability**: Lightweight IDS can be cross-platform, have a small system footprint, and be easily configured by system administrators who need to implement a specific security solution in a short amount of time.

(2) Distributed and Collaborative Architecture (C-dIDS)

The second concept is collaborative distributed IDS with single-level hierarchy (C-dIDS). The dIDS allows computation load and diagnostic responsibilities to be distributed throughout the network. It delegates its responsibilities to a number of distributed components. A number of independent intrusion detection processes monitoring only a small aspect of the IDS are deployed to protect the overall computer infrastructure system. They operate concurrently and co-operate with each other. Moreover, the C-dIDS can provide the foundation for a complete solution to the complexities of real-time detection, while maintaining fault-resistant behaviour. The distributed nature of the data sources allows patterns in the data to be seen that might not be detectable if each of the sources were examined individually. In addition to the above benefits, it will satisfy the following requirements:

**Scalability:** By using a one-level hierarchy dIDS, the detection process will need just less data (compared with other dIDS) to accomplish the cooperation process between different IDS.

**Extensibility:** Each module can be added to or removed from the system without altering other system components, because the intrusion detection processes are independent and thus existing processes do not need to be modified when a new intrusion detection process is added.

**Configurability**: A single intrusion detection process can be simply tailored to the local requirements of a specific host without considering the various requirements of other hosts.

**Reliability:** Our detection process is distributed through four different network levels (layers), and if the intruder is successful in attacking one level, the system will continue applying the other levels of detection. The failure of one local intrusion detection process does not cripple an entire IDS, even though it causes minimal degradation of overall detection accuracy.

**Robustness:** The proposed IDS will be difficult to attack, as it is divided into many detection levels (depending on the number of devices in the network) that make attacking the system much more difficult.

**Flexibility:** The modules will run in parallel and can act independently. Thus, they can be added to and removed from the system without altering other components.

**Minimum system load:** To cooperate between different system IDS modules, each IDS module does not need much transferred information (just one bit of information).

## 1.3 Thesis Organization

The thesis consists of five chapters, the first of which is the introduction. We provide a brief description of Intrusion Detection Systems (IDS), followed by an overview of this thesis' motivations and goals.

In Chapter 2, a brief review of security and IDS is given. We discuss IDS architecture: detection method, analysis techniques, and response components. In addition, we discuss some approaches in IDS such as distributed systems and the Soft Computing (SC) technique. We finish by presenting the current state of the art in IDS and the limitations thereof.

In Chapter 3, we present a lightweight IDS concept. To build a lightweight IDS, we apply two main approaches: the features selection approach for IDS (Fuzzy Enhanced Support Vector Decision Function- Fuzzy ESVDF), and an IDS classification scheme. This chapter is split into two main sections. The first section describes the features selection approach. It starts by briefly reviewing the dimensionality reduction problem for IDS. The proposed algorithm (Fuzzy ESVDF) is then explained, followed by presentation of simulation results and an evaluation of the approach. For the second section, the IDS classification scheme is presented. Essentially, we illustrate the motivation behind the new IDS categorization (classification). We then describe the employed IDS classification scheme while presenting some experimental results. The section closes with some discussion of the approach, and with a conclusion regarding its utility.

In Chapter 4, we present the second concept, a distributed collaborative architecture for IDS and the proposed architecture of the thesis (C-dIDS based on lightweight IDS modules) through two main sections. In the first section, we start by briefly reviewing the distributed IDS. After that, the proposed "collaborative architecture for distributed IDS" (C-dIDS) is outlined. Then the simulation results are presented along with analysis and recommendations. Section 2 of Chapter 4 presents the proposed

architecture (C-dIDS based on lightweight IDS modules). This architecture combines two concepts, which will have been discussed previously in this thesis: lightweight IDS and a distributed collaborative architecture for IDS. To evaluate the C-dIDS, experiments have been carried out and presented. We end the chapter with some conclusions.

Finally, important conclusions and possible extensions to this work are outlined in Chapter 5. We start with a brief review of the thesis' summary and contributions, followed by a discussion of future research directions.

# Chapter 2
# Background and Related Research

This chapter begins with a brief overview on computer security (Section 2.1). We explain the main security elements: security services, security mechanisms, and security policy. A theoretical framework and introduction to Intrusion Detection Systems (IDS) are presented in the next section (Section 2.2). First, a description and evaluation of IDS are given, followed by the presentation of its components, architecture, goals, and functions. Next, an overview of different computer attack categories is given. Then, we briefly review IDS evaluation criteria and different IDS approaches, followed by the major IDS challenges. Finally, the concepts of Soft Computing (SC) and distributed architecture for IDS are discussed in Section 2.3 and Section 2.4, respectively.

## 2.1 Computer Security

According to [129], computer security infrastructure is based on the following three main security services: *confidentiality*, *integrity*, and *availability* in a computer system. Confidentiality is the keeping of sensitive information from unauthorized disclosure, which means that unauthorized parties cannot access information. It is also known as secrecy or privacy. Integrity concerns the protection of sensitive information against unauthorized modifications that are not detectable to authorized users. It provides a mechanism for protecting information against accidents or malicious tampering. Finally, availability is the prevention of unauthorized withholding of information and resources. It is responsible for keeping the computer system working without degradation of access to resources for authorized users when they need it.

Other important security services are *authentication, access control,* and *non-repudiation*. Authentication is the act of verifying the identity of a user logging onto a network. It is the process of determining whether someone or something is, in fact, who or what it is declared to be. Maintaining access control means not only that users can access those resources and services to which they are entitled, but also that they are not denied resources that they may legitimately expect to access. Non-repudiation means that a person who sends a message cannot deny that he sent it and, conversely, that a person who has received a message cannot deny that he received it. In addition to these technical aspects, the conceptual reach of computer security is broad and multifaceted. Computer security draws from such diverse disciplines as ethics and risk analysis, and is concerned with computer crime

(i.e. the prevention, detection, and remediation of attacks), as well as identity and anonymity in cyberspace.

The security services described above provide preventive measures for ensuring the security of the system by helping to avoid security policy violations that can occur. A *security policy* is an organization's statement defining the rules and practices that guarantee confidentiality, authentication, availability, and integrity in a computing system. It plays three major roles: makes clear what to protect and why; it describes the responsibilities for that protection; and it defines the basis on which to recover from damage caused by security breaches. It also regulates how to provide security and handle intrusions. A security policy might include sections on virus detection and prevention, firewall use and configuration, password strength and management, access control rules, physical security, and many others.

*Security mechanisms* are the means for implementing security services. They can be divided into three broad categories: Prevention, Detection, and Recovery.

An *Intrusion Prevention System (IPS)* is the first step in the convergence of networking and security. It provides policies and rules for network traffic along with an intrusion detection system for alerting network administrators to suspicious traffic, as well as allowing the administrator to take action on being alerted. The IPS is not just a perimeter protection element; it delivers its greatest value as a pervasive security element that is deployed at both internal and perimeter network segments.

*Intrusion Detection System (IDS)* is the second line of defense. It gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization). The IDS is designed to detect unscrupulous activities that compromise the confidentiality, integrity, or availability of network or computer systems and to analyze what happens – or has happened – to indicate that the computer has been misused. It does not eliminate the use of a preventive mechanism, but it works as the second defensive mechanism behind a firewall that can monitor the network while leaving network performance unaffected.

*Intrusion Recovery System (IRS)* is the third line of defense. It is comprised of the steps or actions that need to be taken after the system has been compromised, in order to restore it to its previous

condition and avoid further loss from intrusion. It will also terminate intrusion and protect against reoccurrence.

Detection and recovery mechanisms generally involve long-term activities and are necessary because prevention alone can never be wholly adequate. In the following sections of this chapter, we will briefly describe IDS.

## 2.2 Intrusion Detection System (IDS)

An *intrusion* is defined in [127] as any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource by trying to violate a security policy. For example, if a system security policy defines specific authorized users, then the action of sneaking into these users' accounts and transferring these users' files is an intrusion.

*Intrusion detection* is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusion. It aims to protect the confidentiality, integrity, and availability of critical networked information systems by analyzing what happens or has happened during an intrusion, and attempts to locate signs that the computer has been misused. It can also take the appropriate actions to cut off network connections, record events, raise an alarm, and remind system administrators to take proper measures.

*Intrusion detection System* (IDS) is a system that gathers and analyzes information from various areas within a computer or a network to identify attacks made against these components. The IDS uses a number of generic methods for monitoring the exploitations of vulnerabilities. They are useful not only in detecting successful breaches of security, but also in monitoring attempts to breach security, which provides important information for timely countermeasures. Thus, the IDS is useful even when strong, preventive steps are taken to protect computer systems, placing a high degree of confidence in the security it provides [130].

### 2.2.1 Evolution of IDS

James Anderson's paper [131] was the first document to describe the concept of an IDS. The paper described computer security threat monitoring and surveillance. It debated the pros and cons of audit trail data, log files tracking users' access to data, and how the analysis of these documents enabled the reader to detect unauthorized access to data [131].The author identified that the problem with this

system of detection was that the files did not contain enough pertinent data on user access to be used by the security staff reviewing them [131].

Three years later, the first model IDS was developed under the name IDES (Intrusion Detection Expert System). The system was inspired by Dr. Dorothy Denning's paper "An Intrusion Detection Expert Model." The main focus of the paper was that it was possible to create models of users of a system based on the actions of the users on the audit files, and that unauthorized access could be detected by identifying abnormal behavioural patterns in those files [132]. This is the basis for the anomaly-based detection techniques.

The next major IDS was developed at Lawrence Livermore laboratories in 1988 under the name Haystack. Haystack compared audit data to defined patterns of misuse in order to detect intrusions [133]. This was the basis of the signature-based technique for intrusion detection. The next iteration of IDS development was the DIDS (Distributed Intrusion Detection System), where information on client machines and servers was also tracked.

In 1990, Todd Heberlein developed the Network Security Monitor at UC Davis. The Network Security monitor is considered to be the first intrusion detection system. It was mainly used by major government installations where network traffic monitoring was needed. [134] This system applied knowledge of malicious behaviour in general, so it used not only log files but also network packets for detecting patterns which could be malicious [135]. This system generated interesting results, and increased the interest in IDS. With increased interest came increased investment and in the early 90s IDS began to be developed commercially.

Haystack Labs was the first vendor of a commercial IDS with their system called "Stalker." SAIC, another IDS vendor, developed the Computer Misuse Detection System in 1996 This was another successful IDS. The United States Air Force was simultaneously developing an IDS called the Automated Security Incident Measurement (ASIM). This IDS was the first to use software as well as hardware in an IDS. The same group which developed ASIM later left the USAF and founded their own company, Wheel Group, which later released Net Ranger, which was considered to be the first commercially viable IDS. In 1997, Internet Security Systems Co. developed the Real Secure IDS, which is another important IDS entry in the IDS market.

### 2.2.2 Intrusion Detection Architecture

Modern IDS are extremely diverse in the techniques they employ to gather and analyze data. Most rely, however, on a common architecture for their structure, as shown in Figure 2.1:

- *A Detection Model:* This gathers data that may contain evidence of intrusion. All modern IDS monitor host computers, networks, routers or application links to capture intrusion-relevant data.
- *An Analysis Engine:* This can also categorize three types of detections: misuse detection, anomaly detection, and specification detection.
- *The Response Component:* This reports intrusions and takes other responses such as isolation, changing logging or disconnection, etc.

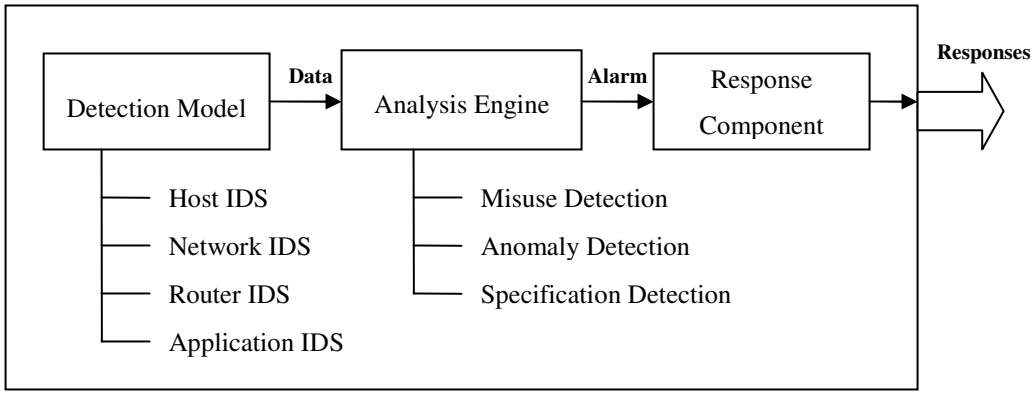



Figure 2.1 IDS Architecture

#### *Detection Model/ Information Sources*

The first distinction can be made in terms of the placement of IDS. In this respect, IDS are usually organized into host-based, network-based, router-based and application-based systems.

a. Host-based Intrusion Detection Systems (HIDS)

Host audit sources are the only way to gather information on the activities of the users of a given machine [32], [123], [136]. Thus, Host Intrusion Detection Systems (HIDS) are present on each host that requires monitoring and collects data concerning the operation of this host. This usually consists

of log files, network traffic to and from the host, or information on processes running on the host. HIDS can determine if an attempted attack was indeed successful and can detect local attacks, privilege escalation attacks, and attacks that are encrypted. However, such systems can be difficult to deploy and manage, especially when the number of hosts needing protection is large. Furthermore, these systems are unable to detect attacks against multiple targets on the network.

b. Network-based Intrusion Detection Systems (NIDS)

Network Intrusion Detection Systems (NIDS) monitor the traffic on the network containing the hosts to be protected and are usually run on a separate machine, called a sensor [9], [13], [29], [31], [137]. NIDS are able to monitor a large number of hosts with relatively little deployment cost and are able to identify attacks to and from multiple hosts. However, they are unable to detect whether an attempted attack was successful, and are unable to deal with local or encrypted arracks.

c. Router-based Intrusion Detection Systems (RIDS)

Router Intrusion Detection Systems (RIDS) enable networks to cooperate in the detection of system attacks and protect the greater network infrastructure [34]. This approach is close to the second approach (NIDS) with a few exceptions. First, a RIDS protects network infrastructure and particularly focuses on routing. Therefore, the target of analysis is mainly on specific protocol traffic instead of general data traffic. Second, a RIDS analyzes the logical behaviour of routing in order to identify the set of states that are indicative of security attacks. These systems ensure a safe, reliable connection between computers over large networks.

d. Application-based Intrusion Detection Systems (AppIDS)

Application Intrusion Detection Systems (AppIDS) is being researched by Robert Sielken and Anita Jones and University of Virginia [148], [149]. Their approach uses the semantics of the application as a further basis for detection of intruders. The AppIDS examines the behaviour of the application: it can observe interaction between the application and the user, and it is able to operate when incoming data is encrypted. However, it is more susceptible to attack, less capable of detecting software tampering, and may be taken in by forms of spoofing.

### *Analysis Techniques*

Once intrusion detection data have been gleaned, IDS uses its analysis to identify intrusions. Three main approaches can be distinguished: misuse detection, anomaly detection, and specification-based intrusion detection, the latter combining misuse and anomaly detection.

a. Misuse detection

Misuse detection attempts to model abnormal behaviour, any occurrence of which clearly indicates system abuse [8], [58], [68]. It aims to discover intrusion by searching for distinguishing patterns or signatures of known attacks. It produces a minimal number of False Positives. Misuse detection can attain high levels of accuracy, but it suffers from many limitations: (1) Difficulty in creating compact models of attacks (models that cover all possible variants of attacks); (2) Inability to detect new intrusions; (3) Signature updating bottleneck; (4) Intrusion variation detection; (5) It is difficult for misused systems to identify attacks that may originate from more than one source, or vary in the means by which they are conducted, or are protracted over long periods of time; (6) Extensive effort is required to construct and maintain a misuse detection system since attack scenarios and system vulnerabilities need to be analyzed and categorized, and the corresponding rules and patterns need to be carefully hand-coded and verified. Misuse detection might be implemented by one of the following techniques: expression matching [50], state transition analysis [50], dedicated languages [50], and burglar alarms [50].

b. Anomaly detection

Anomaly detection attempts to model normal system behaviour, any events that violate this model are considered to be suspicious [18], [48], [51], [117]. It is based on the assumption that intrusion behaviour deviates significantly from previously learned normal behaviour, and employs the user profile as the basis for detection. Any deviation from normal user behaviour is considered an intrusion. Anomaly detection addresses the problem of detecting novel intrusions. However, it suffers from many drawbacks, such as: (1) Inability to identify intrusion, in that it suffers from the problem of how to correctly construct a baseline model of behaviour that is sufficient for complete and correct operation of the system; (2) A higher false alarm rate; (3) Difficulty in determining whether anomalies are caused by intrusions; (4) Concept drifting problem; (5) Mimicry attacks; (6) Intensive computational cost; (7) User behaviour that can change dynamically and can be very inconsistent; and (8) Some intrusions can only be detected by studying the sequential interrelation between events, because each event alone can appear to be normal according to the statistical measures. Anomaly

detection might be implemented by one of the following techniques: statistical models [48], [49], which consist of many different techniques such as threshold measures, mean and standard deviation, and many others [50]; an immune system approach [140]; protocol verification [50]; file checking [50], taint checking [50]; neural networks [29], [117], [33]; fuzzy logic [93]; support vector machine [17]; and data mining techniques [141], [142].

c. Specification detection

This approach was introduced more recently by University of California, Davis, and referred to as specification-based intrusion detection [72], [143], [144], [143] relying on manually setting program behavioural specifications that are used as a basis to detect attacks. It determines whether or not a sequence of instructions violated a specification of how a program, or system, should behave. This technique has been proposed as a promising alternative which combines the strength of misuse-based and anomaly-based detection. Specification-based detection has the potential to provide a very low False Positive rate. It is, however, difficult to model complex programs or systems and write security specifications for them.

### *Response Component*

One major concern is to ensure that in the case of an intrusion attempt, the system is able to detect and to report it [99], [146], [147]. Once the detection is reliable, the next step is to protect the network (responses). In other words, the IDS will be upgraded to an Intrusion Detection and Response System (IDRS). Intrusion responses are a series of actions and countermeasures employed when an intrusion is detected. These actions and measures can prevent further attacks and restore the system to a normal state. Current intrusion response systems can be categorized depending on different criteria such as: degree of autonomy; activity of triggered response; ability to adjust; time response; cooperation ability; and response selection method.

Degree of autonomy is grouped into three categories:

- Notification Response System: notification or alert to the administrator could be the displaying of a pop-up window, or generating an e-mail, pager or mobile phone message.
- Manual Response System: allows administrator to manually launch countermeasures against a detected intrusion by choosing from a predetermined set of responses.

- Automatic Response System: able to choose countermeasures themselves and respond to an attack immediately without human intervention.

Activity of triggered response is grouped into two categories:

- Passive Response System: this is content with merely detecting an intrusion, leaving its handling to a human agent.
- Active Response System: IDS automatically takes action in response to a detected intrusion, reacting immediately to an intrusion as it occurs.

Ability to adjust is grouped into two categories:

- Static: The majority of these systems are static, as the response selection mechanism remains the same during the attack period. These systems can be periodically upgraded by the administrator; such support, however is manual, and often delayed until the moment when a considerable number of intrusions expose the inadequacy of the current response mechanism. Although this approach takes a conservative view of the system and environment, it is simple and easy to maintain.
- Adaptive: The adaptability of the response is the ability of the system to dynamically adjust the response selection to the changing environment during an attack. Adaptation capability can be represented in several ways including (a) adjustment of system resources devoted to intrusion response such as activation of additional IDS, or (b) consideration of success and failure of responses previously made by the system.

Time response is grouped into two categories:

- Proactive: Proactive response systems allow the system to anticipate the incoming intrusion before the attack has affected the resource. Such prediction is generally difficult and often relies on probability measures and analysis of current user or system behaviour.
- Delay: The response action is delayed until the attack has been confirmed. Such assurance may be provided through the confidence metrics of the IDS or the full match of the intrusive trace with an existing attack signature.

Cooperation ability is grouped into two categories:

- Autonomous: Autonomous response systems handle intrusions independently at the level at which they are detected. As such, a host-based IDS detecting an intrusion on a single machine will trigger a local response action such as terminating a process, shutting down the host, etc.
- Cooperative: Cooperative response systems refer to a set of response systems that combine efforts to respond to an intrusion. Cooperative systems can consist of several autonomous systems that are capable of detecting and responding to intrusions locally, though the final, or additional, response strategy is determined and applied globally.

Response selection method is grouped into three categories:

- Static mapping: Static mapping systems are essentially automated manual response systems that map an alert to a predefined response. For example, detecting an attack on a host can trigger the dropping of incoming/outgoing network packets.
- Dynamic mapping: Dynamic response mapping systems are more advanced than static mapping systems as the response selection is based on the certain attack metrics (confidence, severity of attack, etc).
- Cost-sensitive mapping: Cost-sensitive response systems are the only response systems that attempt to balance intrusion damage and response cost. The optimal response is determined based on the cost-sensitive model that incorporates several cost and risk factors.

### 2.2.3 IDS Functions and Goals

Many studies have shown that most computer security incidents are caused by insiders; this results in the need for extra security measures within the organization. IDS may complement other preventive controls (e.g. firewalls) as the next line of defence within the organization. An IDS software or hardware system is placed inside or at the boundary of the protected network to monitor what occurs within the network. It offers the opportunity to detect an attacker who is able to pass through different network devices. Detection can take place at the beginning of the attack, during the attack, or after it has occurred. Once detection is reliable, the next step is to protect the network (responds). The response can be activating an alarm, isolation, changing logging, disconnecting, etc.

The goal of IDS is to accurately detect intrusions, sort out true intrusions from false alarms, and notify network administrators of the activity. Many organizations now use IDS to help them

determine if their systems have been compromised. Given the goal of an IDS, the functions of an IDS can be:

- Monitor and analyze user and system activity
- Audit system configurations and vulnerabilities
- Detect a wide array of intrusions, including outside intrusions and insider attacks, of both known and unknown varieties
- Detect intrusions in a timely fashion
- Present the analysis in a simple, easy-to-understand format
- Achieve a low false alarm rate (high accuracy)
- Inform the system of any suspicious behaviour by sending a report or sounding an alarm
- Assess the integrity of critical system and data files.

An IDS may embody one or more of these functions, depending on the type of IDS, network architecture, and user requirements. Moreover, the combination of these features allows system administrators to more easily handle the monitoring, audit, and assessment of their systems and networks.

### 2.2.4 Computer Attacks Categories

An *intrusion* is defined as any set of actions that attempt to compromise the confidentiality, integrity, or availability of a resource to gain root privilege, whether by exploiting vulnerabilities in the system configuration to access confidential data, or by relying on a legitimate system user to download and run a seemingly legitimate Trojan horse program.

With an increased understanding of how systems work, intruders have become skilled at determining weaknesses in these systems and exploiting them to obtain system privileges and access system resources. Intruders also use patterns of intrusion that are difficult to trace and identify. They frequently employ a series of feints before breaking into target systems and rarely indulge in sudden bursts of suspicious or anomalous activity. They also cover their tracks so that their activity on the penetrated system is not easily discovered.

In general, attack types fall into four main categories [138]:

- Probing: surveillance, among others
- DoS: Denial of Service
- U2Su/U2R: Unauthorized access to Local Super user (root) privileges
- R2L: Unauthorized access from a Remote machine

### *Probing*

Probing is a class of attack where an attacker scans a network to gather information or find known vulnerabilities. There are different types of probes, some of which abuse the computer's legitimate features and others that employ social engineering techniques. This class of attack is the most common and requires very little technical expertise. Examples of this type include IPsweep, Saint, and Satan.

### *DoS Attacks*

Denial of service (DoS) is a class of attack where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine. There are different ways to launch DoS attacks, such as abusing the computer's legitimate features, targeting implementation bugs, or exploiting the system's configuration errors. Examples of this type of attack include DDoS, Pingflood, SYN flood, Mailbomb, and Process Table.

### *U2Su Attacks*

User to root (U2Su) exploits are a class of attack where an attacker starts out with access to a normal user account on the system and is able to exploit a vulnerability to gain root access. Most common exploits in this class of attack are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions. Examples of this type of attack include Eject, Fdformat, Loadmodule, and Perl.

### *R2L attacks*

A remote to user (R2L) attack is a class of attack where an attacker sends packets to a machine over the network, then exploits the machine's vulnerability to illegally gain local access as a user.

Examples of this type of attack include Dictionary, FTP-write, Sendmail, and Xlock.

### 2.2.5 Evaluation Criteria

To evaluate the efficiency of an IDS, there are a number of parameters to be considered [139]:

**Accuracy:** Accuracy deals with the proper detection of attacks and the absence of false alarms. Inaccuracy occurs when an IDS flags a legitimate action in the environment as anomalous or intrusive.

**Efficiency:** An IDS has to perform and propagate its analysis as quickly as possible to enable the security officer to react before much damage has been done, and to prevent the attacker from subverting the audit source or the IDS. The efficiency of the IDS not only encompasses the intrinsic processing speed of the IDS but also the time required to propagate the information and react to it.

**Generality:** An IDS should capture all or almost all known attacks.

**Real Time Performance:** Computer networks have a dynamic nature in the sense that information and the data within them are continuously changing. Therefore, to detect an intrusion accurately and promptly, the system has to operate in real time.

**Robustness:** It should have multiple detection points, which are robust enough against attack and any system faults of the IDS. If intruders already know the existence of an IDS and can subvert it, then the effort to develop the IDS was futile.

**Scalability:** It is necessary to achieve reliable scalability in order to gather and analyze the high-volume of audit data correctly from distributed hosts. In the case of a monolithic IDS, the audit trail collection procedure is distributed and its analysis is centralized. However, it is very difficult to forward all audit data to a single IDS for analysis without losing some of the data. Even if the IDS scales for all audit data correctly, it may cause severe network performance degradation.

**Extendibility:** It should be easy to extend the scope of IDS monitoring by and for new hosts easily and simply regardless of the operating system. When a new host is added to an existing network environment, and especially when this new host runs an operating system that employs a different format of audit data, it is difficult to monitor it in a consistent manner with existing IDS.

**Completeness:** Completeness is the ability of an IDS to detect all attacks. Incompleteness occurs when the IDS fails to detect an attack. This measure is much more difficult to evaluate because it is impossible to have global knowledge about attacks or abuses of privileges.

Even though various approaches have been developed and proposed, no existing IDS satisfy all of these requirements completely.

### 2.2.6 IDS Approaches

There has been steadily growing interest in research and development of IDS. The main goal was to create a system capable of detecting different kind of attacks. To accomplish this goal, researchers have been exploring various tools and techniques such as Pattern Matching [58], [59], Statistical Models [48], [49], State Transition Analysis Technique [60], Information Theoretic Measures [51], Intrusion Correlation (Data Mining) [74], [75], [76], Immune System [52], [53], [54], File Checking [78], Whitelisting [79], Colored Petri nets [81], etc.. The next paragraphs survey some of these approaches, and give examples of currently available tools using them. No intrusion detection approach stands alone as an ideal system which captures all attack types; each approach is technically suited to identify a subset of security violations. The intent of this sub-section is to give a brief overview of current intrusion detection techniques, to better identify how our proposed system (C-dIDS) fits into the general scheme of things. Understanding the strengths and limitations of these approaches will clarify the benefits, as well as the tradeoffs, to the approach presented in this thesis.

Pattern Matching [58], [59], is the simplest technique used for anomaly IDS. This technique searches an event stream for occurrences of specific patterns. Although this technique is fast, it requires an understanding of the nature of the attack, which implies that human experts must work on the analysis and representation of the attacks. This tends to be time-consuming and error-prone. Moreover, this technique suffers from scalability issues, either in terms of speed or the number of patterns to be searched, primarily due to limited and expensive logic resources. Only those attack scenarios which are known and constructed into patterns by the system can be detected. Attacks involving spoofing, and passive methods of attack like wire-tapping cannot be detected.

Statistical Modeling [48], [49] is among the earliest methods used for anomaly detection in electronic information systems. It measures the user and system behaviour by a number of variables sampled over time, and builds profiles based on the variables of normal behaviour. The actual variables are then compared against the profiles, and deviations are considered abnormal. There are many statistical techniques such as threshold measures, mean and standard deviation, Markov process model, clustering analysis, etc [50]. While these statistical techniques have some value, they are insensitive to the order of the occurrence of events, which causes them to miss the sequential interrelationships between events. For intrusions reflected by such an ordering of patterns, a statistical

IDS will miss these intrusions. Moreover, this approach requires the construction of a model for normal user behaviour, and any user behaviour that deviates significantly from this normal behaviour is flagged as an intrusion. It can also be difficult to determine the correct anomaly threshold at which behaviour is to be considered an intrusion. Also, to apply statistical techniques, one has to assume that the underlying data comes from a quasi-stationary process, which may not always hold.

State Transition Analysis Technique (STAT) [60] is one of the famous rule-based expert systems for detecting penetrations. It was developed by the Reliable Software Group at UCSB for misuse detection in UNIX systems, distributed systems, and networks. The STAT uses the state transitions of the system to identify intrusions. This method constructs the state transition diagram, which is the graphical representation of intrusion behaviour as a series of state changes that lead from an initial secure state to a target compromised state. State transition diagrams list only the critical events that must occur for the successful completion of the intrusion. The main advantage of this technique is that it allows a complex intrusion scenario to be modeled in a simple way, and is capable of detecting slow, distributed, and cooperative attacks, variations to known attacks, and attacks which span across multiple user sessions. Too, it improves the ability to automatically determine the data to be collected to support intrusion analysis. This enables a lightweight and scalable implementation of the network probes. On the other hand, it may have difficulty in expressing the attacks scenarios. Also, it can only construct patterns from sequences of events, not from more complex forms, and therefore some attacks cannot be detected as they cannot be modeled with state transitions.

Information theoretic measures is another technique that has been used by many researchers for IDS [51], [62], [80]. This technique computes information content in data using information theoretic measures such as entropy, conditional entropy, relative conditional entropy, information gain, and information cost, and uses them to describe the characteristics of audit data and to build anomaly detection models. It operates in an unsupervised mode. It requires, however, an information theoretic measure sensitive enough to detect irregularity induced by very few outliers.

Data mining generally refers to a process of non-trivial extraction of implicit, previously unknown, and potentially useful information from databases. The key concepts of using data mining in IDS are to discover consistent and useful patterns of system features that describe user behaviour, and to use the set of relevant system features to compute classifiers which can recognize anomalies and known intrusions [36], [74], [75], [84], [142]. These data mining techniques have been garnering increasing research interest, since they can automatically discover detailed attack or normal models that can be

easily understood by human beings. However, these techniques tend to generate a large number of models, especially for large inputs of data. In addition, it requires extra human intervention to reduce and refine the extracted models.

Many immune systems have recently been developed for IDS [52], [53], [54], [76]. In the immune system approach, applications are modeled in terms of sequences of system calls for a variety of different conditions: normal behaviour, error conditions, and attempted exploits. Comparing this model to observed event traces allows classification of normal or suspicious behaviour. In general, this technique provides the computer system with a high level of protection from a specific number of attacks in a robust, autonomous, adaptive, self-organization and distributed manner. However, it cannot detect attacks based on race conditions or policy violations. Moreover, it faces other difficulties, such as its inability to efficiently map the entire non-self universe, its definition of self-ambiguous, and self/ non-self changes over time.

Dedicated languages are the most widely used approach misuse detection. Each attack signature takes the form of a specialized program, with row events as input. Any input triggering a filtering program, or input that matches internal alert conditions, is recognized as an attack. Unfortunately, there is no common language for describing attacks. In general, there are six different classes: event languages [55], [56], [62], response languages, reporting languages [63], [64], correlation languages [65], [66], [68], exploit languages [69], [70], and detection languages [65], [66], [68], [71], [72], [73]. These language classes define different scopes and goals. While the dedicated languages technique offers great flexibility in matching attack scenarios, it needs significant understanding of protocols, attacks involved, and programming ability. Moreover, attacks with a signature variations string may not be captured.

Nevertheless, there are common challenges in the current studies of IDS which are reflected on IDS performance, such as high False Positive/Negative, limit scalability, etc. More detail about these challenges is discussed in the following sub-section.

### 2.2.7 Major IDS Challenges

Considering the surveyed literature, it is clear that the current view of IDS is that it is far from a reliable protective system. This sub-section briefly identifies some of the inherent characteristics that limit the performance of the different IDS techniques. They are as follows:

- *High False Positives*

  False Positives are those sequences of innocuous events that the IDS classifies as intrusion. A common complaint is that the large number of False Positives generated by ID systems makes it hard to filter out false attacks without potentially missing true attacks. Another crucial problem that arises from a high number of False Positives is related to incident handling; that is, security administrators are uncertain how to respond to mitigate the risks if a certain degree of accuracy cannot be achieved. There is no decision rule associated with each alert to tell the security administrator whether he should ignore the alert or simply terminate the suspicious session.

- *High False Negatives*

  False negatives refer to intrusion attempts that the IDS fails to report.

- *Limited Scalability*

  It is very difficult to forward all audit data to a single IDS for analysis without losing the data. Even if it scales for all audit data correctly, it may cause severe network performance degradation.

- *Lack of Context Information*

  Anomaly detection fails to provide adequate contextual information for the security administrator in locating the attack. This weakness increases the difficulties of alert handling.

- *Too Many Variants*

  Because of newly merging attack behaviours and quickly spreading malicious code, it is very difficult to determine the nature of an event before significant damage has been done. Another affliction is that the exploit codes targeting known vulnerabilities do not stay unchanged forever. (If the computers can provide enough decision-supporting analysis reports, then the system administrators can more easily determine the correct action to take in a crisis.)

- *Writing Signatures for IDS is a Very Difficult Task*

  In some cases, the appropriate balance between an overly specific signature (which is not able to capture all attacks) and an overly general one (which recognizes legitimate actions as intrusions) can be difficult to determine.

- *Skewed Class Distribution*

  The training set consists of many normal examples and a small number of attack examples – an imbalance between these two data types may cause difficulties in recognizing the correct patterns.

- *Propagation of Number of Attacks*

  The rapidity of intruder tool improvement increases the number of attacks and strategies that are used to attack the system.

- *IDS maintenance*

  Like any other system, maintenance must be performed. In misused systems, signatures must be updated at regular intervals, an onerous task in most cases.

- *Other Difficulties*

  One of the major difficulties is that some actions can be normal in certain environments but may be malicious in others.

## 2.3 Soft Computing Approaches for IDS

The application of Soft Computing (SC) is widely used for IDS because of its features, such as accuracy (low False Positive and False Negative rates), flexibility (not easily fooled by small variations in intrusion patterns), adaptability in new environments (modular with both misuse and anomaly detection components), low solution cost, real-time performance (fast recognition and ability to classify different attacks), and ability to generalize from learned data. The SC is the general term for describing a set of optimization and processing techniques that are tolerant of imprecision and uncertainty. The ability of SC tools to deal with uncertain and partially true data makes them suitable for application in IDS. The SC is used to create a system of detecting and characterizing anomalous network behaviour. The principal constituents of SC techniques are *Neural Networks (NN), Fuzzy Logic (FL)*, and *evolution computation*, as shown in Figure 2.2.

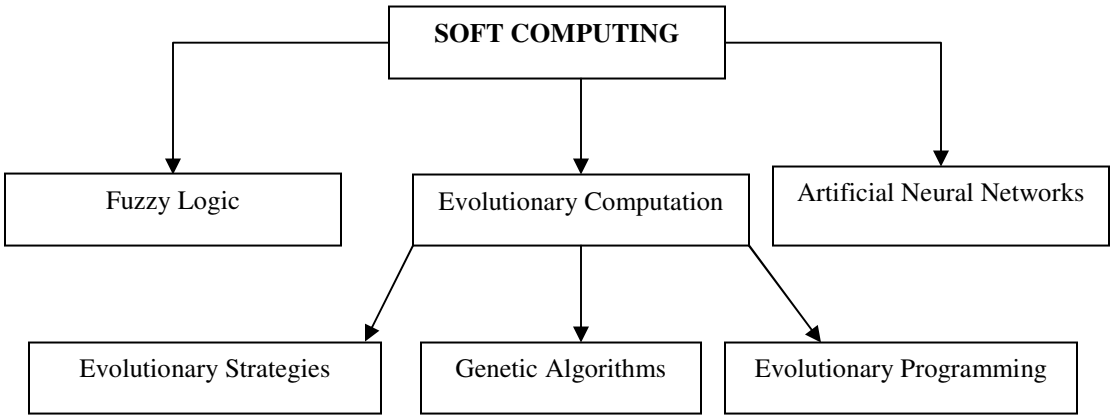



Figure 2.2 Soft Computing Diagram

FL systems [6] are useful in situations when human expertise (expertise that cannot be translated into a set of equations) needs to be incorporated into the decision-making process. Evolutionary programming, evolutionary strategies, and genetic algorithms [150] are useful for optimization problems whose particular difference is how they avoid local extremes. Finally, Artificial Neural Networks (ANNs) [7] are useful when complex relationships (or patterns) in data need to be extracted. ANNs are tolerant of imprecise data and uncertain information: with their ability to generalize from learned data, they seem to be an appropriate approach to IDS.

Based on IDS, most of the work conducted in the literature falls into two key areas: detection model and generation, and intrusion features selection. For detection model and generation, numerous SC techniques are adopted to build efficient detection models such as FL [11], [12], GA [13], [14], [15], NN [8], [9], [10], [16], and Support Vector Machines (SVM) [17], [18]. For intrusion features selection, much research has tried to select the important intrusion features using different SC approaches, such as NN [19], [20], GA [21], [22], [23], [24], SVM [19], [25], [26], [34], and other optimization tools [27], [28].

The rest of this section briefly introduces the various SC techniques, such as FL, GA, NN, and SVM approaches in both areas of IDS: detection model and generation, and intrusion features selection. In addition, we mention some related works for each of these approaches.

FL [6] is a mathematical technique for dealing with imprecise data and problems with many solutions. FL works with ranges of values, solving problems in a way that more resembles human

logic. FL is often used in systems where state transitions should be softened when making decisions with fuzzy boundaries. FL has been used in both IDS research areas: detection model and generation, and intrusion features selection. For the detection model and generation area, many researchers have proposed the application of FL such as Piyakul *et al* [11] and Zhang *et al* [12]. Dickerson *et al* [37] proposed Fuzzy Intrusion Recognition Engine (FIRE), which is a network intrusion detection system that uses fuzzy systems to assess malicious activity against computer networks. The system uses an agent-based approach, and each agent performs its own fuzzification of input data sources. At the end, all agents communicate with a fuzzy evaluation engine that combines the results of individual agents using fuzzy rules to produce alerts that are true to a degree. For the intrusion Features Selection area, Xin *et al.* [82] uses interactive data visualization to analyze the features of several different intrusion detection scenarios. Visualizing the data helps to find the most import features that are used to identify intrusions and if they can be characterized as fuzzy sets or by Boolean variables. These features can then be input into a fuzzy cognitive map that serves to fuse the inputs to detect more complex attacks. Most fuzzy approaches in this area are integrated with other SC methods [84], [85], [86].

GAs are a family of computational models based on principles of evolution and natural selection. These algorithms convert the problem in a specific domain into a model by using a chromosome-like data structure and evolve the chromosomes using selection, recombination, and mutation operators. In the IDS detection model, GA can be used to evolve simple rules for network traffic. These rules are used to differentiate normal network connections from anomalous connections. Recent researchers [13], [14], [15] have demonstrated that the GA field is an emerging field in computer security, especially with regard to IDS detection models. Moreover, GAs have been used as one aspect of the IDS features selection approach [21], [22], [23], [24]. Shazzad *et al.* [21] proposed a hybrid features selection method by combining Correlation-based Features Selection (CFS), SVM, and GA. The GA is used to generate subsets of features from the given features set, which is then evaluated by CFS and SVM to pick the best features set. They combined three different approaches and were able to reduce the number of features from 41 to 12 for the DARPA dataset. Alexander *et al.* [22], [24] set out GA that performs the tasks of features selection and architecture optimization for Radial Basis Function (RBF) networks. Also, Kim *et al.* [23] proposed a features selection method identical to the previous method, but they used GA techniques to obtain the optimal features set and the optimal parameters for a kernel function of SVM.

ANNs have been extensively used to detect both misuse and anomaly patterns. ANNs are algorithmic techniques [6], [7] used to first learn the relationship between two sets of information and then generalize to obtain new input-output pairs in a reasonable way. The ANNs consist of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. They are widely considered an efficient approach to adaptively classify patterns due to their capability to compact knowledge representation, even if the data are complex or non-deterministic. This capability makes them an effective implementation model for an IDS. Researchers have proposed several approaches in this regard in order to improve IDS accuracy [29], [30], [31], [32]. Jian *et al.* [33] use backpropagation (BPL) neural networks to detect anomalous user activities. They have shown that NN can be used successfully as a method for training an IDS and providing it with learning skills. Also, Lilia *et al.* [8] present a network detection method using a Hamming net, which is a type of NN with special properties that make it suitable for real-time classification. Moreover, Chunlin *et al.* [16] proposed two hierarchical IDS frameworks using Radial Basis Functions (RBF). They used a serial hierarchical IDS (SHIDS) to identify accurately misuse attacks and adaptively identify anomaly attacks, and then used parallel hierarchical IDS (PHIDS) to enhance SHIDS functionality and performance. For the other IDS research area, features selections, Sung *et al.* [19] exploited SVM and NN to categorize and identify features based on some performance criteria by ranking feature importance for each kind of attack, such as probe, DoS, R2L, and U2R. This approach is based on deleting one feature at a time; the resultant data set is then used for training and testing of the classifier (either NN or SVM). Then the classifier's performance is compared to that of all features based on performance criteria. Finally, the important feature is ranked according to a set of rules based on the performance comparison. Glovko *et al.* [20] proposed NN architectures for the IDS. The proposed approach is based on combining two different NN: Principle Component Analysis (PCA) and Multilayer Perceptron (MLP). PCA (linear and nonlinear PCA) networks are employed for important data extraction and high dimensional data vectors reduction. MLP is employed to detect and recognize attacks using extracted-features data instead of original data.

Recently, SVMs have been used to detect intrusion due to their good generalization characteristics and ability to overcome the curse of dimensionality. SVM is a statistical machine learning algorithm that maps input (real-value) feature vectors into a higher dimensional feature space through nonlinear mapping. The SVM is primarily a classier method that performs classification tasks by constructing linear classifying (hyperplanes) in a multidimensional space that separates cases of different class

labels. A special property of SVM is that they simultaneously minimize the empirical classification error and maximize the geometric margin by using a quadratic optimization problem with bound constraints and one linear equality constraint. There are two other key concepts of SVM: Soft Margin and Kernel concept. SVM are powerful tools for providing solutions to classification, regression, and density-estimation problems. Kun *et al.* [18] proposed an approach to intrusion detection using SVM for anomaly detection. It is a one-class SVM-based approach, which delivers a highly accurate rate on the testing set. In addition, John *et al.* [17] proposed using the SVM learning approach to classify network requests. They employed a new method – ArraySVM – and by their experiments showed satisfactory system performance in terms of training time and accuracy. For IDS features selection, Mukkamala *et al.* [25], [26], [34] proposed a router-based approach to detect DoS attacks by using SVM. They identify DoS-pertinent features by using Support Vector Decision Function (SVDF) and evaluate the applicability of using these features in the detection of online novel DoS attacks on a performance network. SVDF calculates the weight of the features to rank their significance. For example, in the equation features.

$$F(X) = \sum_{i=1}^{n} W_i X_i + b \tag{1}$$

where $W$ is the weight vector , $b$ is a bias value, and $n$ is the number of features. They ranked each feature depending on the value of its weight. The features with large weight values are considered to be the features of the greatest effect (important features) and are used for the detection process. In [26], they used eleven features as important features for the detection process. In [34], they claimed that using six important features from among the eleven features can give excellent performance.

Several other Soft Computing (SC) techniques are used to improve the performance of the IDS and much work has been done in this area. For example, GAO *et al.* [27] proposed Ant Colony Optimization (ACO) and Srinoy [28] proposed Particle Swarm Optimization (PSO) to select the best features set for IDS. Chen *et al.* [35] proposed Flexible Neural Tree (FNT) to identify important input features in building an IDS that is computationally efficient and effective. The FNT structure is developed using an evolutionary algorithm, and the parameters are optimized by a particle swarm optimization algorithm. Li *et al.* [36] proposed a supervised clustering and classification algorithm (CCAS) for IDS. This algorithm utilizes a heuristic in grid-based clustering. Several post-processing techniques including data redistribution, supervised grouping of clusters, and removal of outliers are used to enhance the scalability and robustness. Zanero *et al.* [38] proposed a two-tier architecture for IDS: the first tier is an unsupervised clustering algorithm which reduces the network packets payload

to a tractable size. The second tier is a traditional anomaly detection algorithm, whose efficiency is improved by the availability of data on the packet payload content. Wang *et al* [39] proposed a new clustering algorithm, FCC, for IDS based on the concept of fuzzy connectedness. This approach starts with a single or a few seed points in each cluster, and all the data points are dynamically assigned to the cluster that has the highest fuzzy connectedness value (strongest connection).

## 2.4 Distributed Intrusion Detection Systems

With the increasing connectivity and complexity of heterogeneous computer systems, it is likely unrealistic to expect that an IDS should be capable of correctly classifying every event that occurs on a given system. In addition, there are the limitations of a centralized IDS, such as: a single point of failure; limited scalability; frequent overload; vulnerability to subversion; and difficulty in configuring or adding capability to the IDS. An IDS should consist of multiple entities working independently to cover the huge amount of data and traffic in the system, and should allow changes to these entities without any modifications made to other entities; this is accomplished by using an IDS with distributed architecture. Distributed IDSs (dIDSs) are based on distributed IDS entities located on different locations within the network, which monitor separately and communicate and cooperate with each other. The dIDS allows computation load and diagnostic responsibilities to be distributed throughout the network. It can provide the foundation for a complete solution to the complexities of real-time detection, while maintaining fault tolerance behaviour. It allows early detection of planned and coordinated attacks, thereby allowing network administrators to take preventive measures. dIDS also helps to control the spreading of worms, improves network monitoring, incident analysis, attack tracing and so on. Also, it has scalability to detect general attacks or a specific attack, in addition to providing significant advantages in flexibility, extendibility, and resistance to compromise.

A number of dIDS have been proposed for a distributed environment. Early systems included DIDS (Distributed Intrusion Detection System) [41], NADIR (Network Anomaly Detector and Intrusion Reporter) [45], CSM (Cooperative Security Managers) [46], GrIDS (Graph-based Intrusion Detection System) [42], EMERALD (Event Monitoring Enabling Response to Anomalous Live Disturbances) [43], AAFID (Autonomous Agents for Intrusion Detection) [44], CIDF (Common Intrusion Detection Framework) [156] and MAIDS (Mobile Agent Intrusion Detection System) [47]. The rest of this section briefly introduces some of these projects.

DIDS [41] incorporates Haystack and NSM (Network Security Monitor) in its framework. This system requires the audit data collected from different places to be sent to a central location for

analysis. The DIDS operates on a local area network (LAN) and consists of three major components: the host monitor, the LAN monitor, and the central manager. Each host is monitored by a host manager. This manager is a collection of processes running in the background of the host. Also, each LAN is monitored by a LAN manager, which operates just like a host manager except that it analyzes LAN traffic. Finally, there is a central manager which is placed at a single secure location and controls the entire system. This central manager receives reports from various host and LAN managers, and by processing and correlating these reports, it detects intrusions. The DIDS itself is not fully distributed because it relies on both distributed and centralized resources to detect intrusions. This technology faces a number of challenges such as its centralized nature, arbitrary definitions of abnormal activities, and ineffective coordination between the DIDS modules.

The NADIR system [45] performs distributed data collection by employing the existing service nodes in the Los Alamos National Laboratory's Integrated Computer Network (ICN) to collect audit information. The NADIR examines the network traffic at the service and protocol level by using a statistics-based anomaly detector and an expert system, which is then analyzed by a central expert system. The major drawback of NADIR is its centralized analysis, which severely limits the scalability of the detection algorithm. Moreover this system, NADIR, would not easily be ported to an internetworked environment with many heterogeneous systems.

The CSM [46] are employed to perform dIDS that does not need a hierarchical organization or a central coordinator. Each individual CSM detects malicious activity on the local host. When suspicious activity is detected, each CSM will report any noteworthy activity to the CSM on the host from which the connection originated. The local CSM will not notify all networked systems, but rather only the system immediately before it in the connection chain. The architecture of the system allows for CSM to take reactive actions when an intrusion is detected. Unclear aspects are the mechanisms through which CSM can be updated or reconfigured, and the intrusion detection mechanisms that are used locally by each CSM.

GrIDS [42] uses graph engines that build a graph representation of activity in the network to detect possible intrusions. It aggregates computer and network information into activity graphs which reveal the casual structure of network activity. The GrIDS is able to detect large-scale automated and spreading attacks. Also, it facilitates reporting, policy statements, and process rules. It provides mechanisms to allow third-party security tools to be used as data sources. On the other hand, the judgment of intrusions still needs human input in order to complete.

EMERALD [43] is intended as a framework for distributed, interoperable computer and network intrusion detection. It employs entities called service monitors that are deployed to hosts and perform monitoring functions. They define several layers of monitors for performing data reduction in a hierarchical fashion. Monitors can be programmed to perform any function. However, this model does not scale well for large networks. The large number of events and devices distributed across the network can generate too much network traffic and too much data to be stored in one location efficiently. It also does not cover distributed services (e.g., DNS, firewalls).

AAFID [44] is a distributed intrusion detection architecture and system, developed in CERIAS at Purdue University. It is agent-based, employs a hierarchical structure and the data are collected and analyzed locally. Nevertheless, there is still a highest-level entity in the AAFID architecture, which is the bottleneck of this system and leads inevitably to the matter of a single point of failure. Also, if the two or more IDS that are far part in the hierarchy detect a common intruder, the two detections cannot be correlated until the messages from the different IDS reach a common high-level IDS. This will require the messages to traverse multiple IDS resulting in communication overheads. In addition, it has limited scalability, performance, user interface and security.

CIDF [156] was an effort to standardize intrusion detection to some degree by enabling different intrusion detection and response components to interoperate and share information and resources in a distributed environment. The intrusion detection inter-component adaptive negotiation protocol helps cooperating CIDF components to reach an agreement on each other's needs and capabilities.

MAIDS [47] are also typical distributed IDS. It is an end-to-end procedure for intrusion detection. Known vulnerabilities of a system are expressed in an abstract "Software Fault Tree" (SFT) form, then converted to a Colored Petri Net (CPN), and finally into a system of independent agents. These systems suffer from a number of problems such as a lack of an effective coordination mechanism to detect a complicated attack, and the security of the system itself is almost unconsidered.

The research on dIDS [1], [2], [3], [4], [5], [155] is a rapidly growing area of interest because the existence of dIDS techniques is increasingly unable to protect the global distributed information infrastructure. So, the existing dIDS must be updated and improved constantly to adapt to the ever-changing environment and they should be studied in greater depth in order to ensure better system security.

## 2.5 Conclusion

In this chapter, we presented a brief review of IDS (evolution, architecture and components, goals and functions), followed by presenting the current approaches for IDS such as Pattern Matching, Statistical Models, State Transition Analysis Technique, Information Theoretic Measures. Given the shortcomings of current IDS, our research focus is on combining two main concepts to improve the performance of IDS. The first concept is using lightweight IDS modules. To build a lightweight IDS module, we use two approaches: features selection approach, and an IDS classification scheme. The first approach depends on Soft Computing (SC) to select the appropriate features set for IDS. SC is the general term for describing a set of optimization and processing techniques that are tolerant of imprecision and uncertainty, and that make them attractive to be applied in IDS. The second approach is the IDS classification scheme. This novel scheme employs multiple specialized detectors in each layer of the network TCP/IP network model, which helps in the collection of efficient information. This increases system efficiency and reduces the system's scalability. The second concept used in this thesis proposes a distributed collaborative architecture for the IDS. This architecture can be useful for efficiently designing and maintaining secure networks; each module operates cooperatively yet independently, providing for efficient, real-time response and distribution of resources.

The proposed system, Collaborative Distributed Intrusion Detection System (C-dIDS) based on lightweight IDS modules, combines two concepts: the SC approach to build a lightweight IDS, and the dIDS approach with a novel architecture. A detailed description for each of these approaches is given in Chapter 3 and Chapter 4 respectively.

# Chapter 3
# Lightweight IDS

The intrusion detection system deals with huge amounts of data, which can contain irrelevant and redundant features. This can cause a slow training and testing process, higher resource consumption, and a poor detection rate. Therefore, using a lightweight IDS is an important issue in intrusion detection. Lightweight IDSs are small, powerful, and flexible enough to be used as permanent elements of the network security infrastructure. They should be easily configurable by system administrators who need to implement a specific security solution in a hurry. Also, they should be able to be easily incorporated into any network security architecture with minimal disruption to operations.

Building a Lightweight IDS is the first goal of this thesis, in order to improve the performance, scalability, generality, and extensibility of IDS. Most current work builds a lightweight IDS by only applying one features selection approach, which is usually consider to be inefficient. In our case, however, we will use two different approaches to achieve a lightweight IDS.

The first approach uses a features selection approach. We will apply a novel algorithm for features selection based on a Support Decision Function (SVDF) and Forward Selection (FS) approach, with a fuzzy inferencing model called Fuzzy ESVDF [118], [119]. The Fuzzy ESVDF is able to significantly decrease training and testing times while retaining high detection rates with low False Positive rates.

The second approach uses a new IDS classification scheme. The IDS classification scheme divides the detection process into four types according to the TCP/IP network model (Application Layer, Transport Layer, Network Layer, and Link Layer). This IDS classification can enhance an organization's ability to detect most types of attack (i.e., it improves system accuracy and generality). Also, it can improve system scalability in reducing the amount of data (features) needed to accomplish the detection process.

This chapter is split into two main sections. Section 3.1 describes the features selection approach, while Section 3.2 describes the IDS classification scheme. Our conclusion is drawn in Section 3.3.

Section 3.1 presents the features selection approach for an IDS. Basically, it begins with a brief overview of the dimensionality reduction problem, and then demonstrates the proposed approach (Fuzzy ESVDF), followed by experimental results and discussion. Finally, summary is drawn.

Section 3.2 describes the IDS classification approach. It starts by providing an overview of the TCP/IP model with attack classification, followed by the motivations behind this new IDS classification scheme. After that, we present the proposed approach with some experiments and results, ending with discussion and summary.

## 3.1 Features Selection Approach

One key problem which arises in a wide variety of fields, including pattern recognition and machine learning, is the so-called "feature selection". In complex classification domains, some features may be redundant and/or irrelevant. Extra features can increase computation time, and can have an impact on system accuracy. Features selection improves classification by searching for the subset of features which best classify the training data. Accordingly, features selection is considered to be a very important issue in IDS in achieving maximal performance. In this section, we introduce a novel algorithm for features selection based on a Support Vector Decision Function (SVDF) and a Forward Selection (FS) approach with a fuzzy inferencing model called Fuzzy ESVDF. This is the first approach to build a lightweight IDS, with the goal of improving IDS' performance in terms of accuracy and efficiency (training time and testing time) [118], [119].

### 3.1.1 Dimensionality Reduction

Dimensionality reduction [87], [88] is an important topic in machine learning. Elimination of useless (irrelevant and/or redundant) features [90] enhances the accuracy of the classification while speeding up the computation. It simplifies the classification by searching for the subset of features which best classifies the training set, and allows the extraction of easily interpretable rules, thus improving the overall performance of the classifier and overcoming many problems, such as the risk of "overfitting". Moreover, it helps us to understand the data, and reduces the measurement and storage requirements [91].

Current dimensionality reduction methods can be categorized into two classes: features extraction and features selection. Features extraction [92], [93] involves the production of a new set of features from the original features in the data, through the application of mapping. The dominant features extraction techniques are Principle Component Analysis (PCA) [94] and Linear Discriminant Analysis (LDA) [95]. In contrast, features selection [96], [97], [100], [101], [102], [103] selects the "best" subset of the original features. It reduces the number of features and removes irrelevant, redundant, or noisy data. In terms of features selection, several researchers have proposed identifying

important features through wrapper and filter approaches [90] [104]. The wrapper method [22], [26], [34], [65] exploits a machine learning algorithm to evaluate the fitness of features or a feature set. It provides better performance in the selection of suitable features, since it uses the performance of a learning algorithm as an evaluation criterion. The most widely employed wrapper methods are Forward Selection (FS) [105], Backward Elimination (BE) [105], and Genetic search [106].

In contrast, the filter method doesn't use a machine learning algorithm to filter out irrelevant and redundant features; instead, it uses the underlying characteristics of the training data to evaluate the relevance of the features (or feature set) by several independent measures, such as distance measures, correlation measures, and consistency measures [107], [108]. The most widely employed techniques in this area are Relief [109] and Focus [110]. In general, wrapper approaches demand heavy computational resources, but they can achieve better results than filters because they are tuned to the specific interaction between an induction algorithm and its training data. However, they tend to be much slower than feature filters because they must repeatedly call the induction algorithm and must be re-run when a different induction algorithm is used.

On the whole, since the elimination of insignificant and/or useless inputs leads to a simplified problem and possibly a faster and more accurate classification, features selection is considered to be a very important issue in IDS in order to achieve maximal performance [151], [145]. Features selection can improve the generalization performance of intrusion detection and make the detection more time efficient. Faster training and testing helps to build lightweight IDS and provides ease of maintenance or modification of an IDS. Furthermore, a small number of input features lead to a reduction in execution times, which is important for on-line detection of attacks.

### 3.1.2 Fuzzy ESVDF Approach

We propose a new features selection approach called Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF) based on a Support Vector Decision Function (SVDF) and Forward Selection (FS) with a fuzzy inferencing model [118], [119]. The Fuzzy ESVDF is an iterative algorithm, where each iteration consists of two steps: feature ranking and feature selecting. In feature ranking, SVDF is evaluated to rank each specified candidate feature. Then in feature selecting, FS is applied with the fuzzy inferencing model to select the features according to a set of fuzzy rules based on a comparison of performance. As shown in Algorithm 3.1, the algorithm starts by picking three features from the features set (S1) with the highest weight values (S1 contains all the features with weight values equal to or greater than one; the weight value is calculated by SVDF (1)) and putting

them in the features set (S2), then calculating the classification accuracy and training time for S2. The feature with the next highest weight value from S1 is added to S2 while calculating their performance metrics. Through this process, two types of comparisons are made: a local fuzzy comparison and a global fuzzy comparison. The local fuzzy comparison compares the performance of S2 with the performance from the previous iteration. If the first value is less than the second value, the added feature is ignored; otherwise, it is kept in S2. In the global fuzzy comparison, the classification accuracy of S2 is compared with the global accuracy, which is equal to the minimum of two values: the accuracy of all the features and the accuracy of S1. If the classification accuracy of S2 is equal to or greater than the global accuracy value, the algorithm will stop and S2 will be the selected features set; otherwise, it will continue execution.

The local fuzzy comparison is ranked according to a fuzzy system that takes two inputs: the percentage of increase or decrease in training time as one input, and the percentage of increase or decrease of accuracy as the second input. It compares the performance of the current value with the performance of the previous. The first and the second input variables (percentage of change in the training time and accuracy) are represented by three fuzzy sets: "increase," "same," and "decrease" with their corresponding membership functions, as shown in Figure 3.1. "Increase" refers to the case where the percentage of change (accuracy and time $_{\text{calculated by current selected features}}$ − accuracy and time $_{\text{calculated by previous selected features}}$) in the training time and accuracy is slightly positive. This means that the training time and accuracy slightly increase after a feature is added. "Same" refers to the case where the training time and accuracy remain almost the same. The system has one output ranging from "0" to "1" where "0" represents a non-important feature and "1" represents an important feature in the detection process.

The knowledge base is implemented by means of "if-then" rules. Nine rules are needed to describe the system and rank each feature as "important" or "non important," according to the following rules:

1. ***If*** training time decreases ***and*** accuracy decreases, ***then*** the feature is non-important
2. ***If*** training time decreases ***and*** accuracy does not change, ***then*** the feature is important
3. ***If*** training time decreases ***and*** accuracy increases, ***then*** the feature is important
4. ***If*** training time does not change ***and*** accuracy decreases, ***then*** the feature is non-important
5. ***If*** training time does not change ***and*** accuracy does not change, ***then*** the feature is important
6. ***If*** training time does not change ***and*** accuracy increases, ***then*** the feature is important
7. ***If*** training time increases ***and*** accuracy decreases, ***then*** the feature is non important
8. ***If*** training time increases ***and*** accuracy does not change, ***then*** the feature is non-important

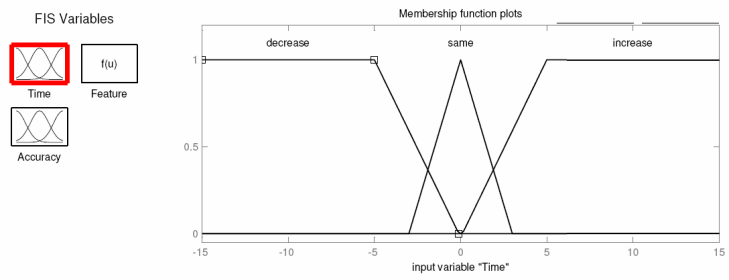
9. ***If*** training time increases ***and*** accuracy increases, ***then*** the feature is non-important

The global fuzzy comparison compares the classification accuracy of S2 with the global accuracy. The comparison is ranked according to a fuzzy system that takes only one input variable (percentage of change in accuracy). This input variable is represented by three fuzzy sets: "increase," "same," and "decrease" with their corresponding membership functions, as shown in Figure 3.2. "Increase" refers to the case where the percentage of change (selected features set accuracy – global accuracy) in accuracy is slightly positive. This means that the training accuracy slightly increases after a feature is added. "Same" refers to the case where there is no change in accuracy. The system has one output ranging from "0" to "1", where "0" represents a loop to continue and "1" represents a loop to stop.
The knowledge base is implemented with three "if-then" rules. Only three rules are needed to describe the system and decide whether to continue adding features:

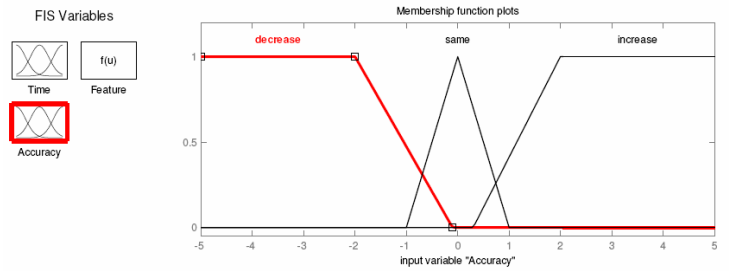
1. ***If*** accuracy increases, ***then*** stop adding features
2. ***If*** accuracy does not change, ***then*** stop adding features
3. ***If*** accuracy decreases, ***then*** continue adding features

**Algorithm 3.1** The Fuzzy ESVDF Algorithm

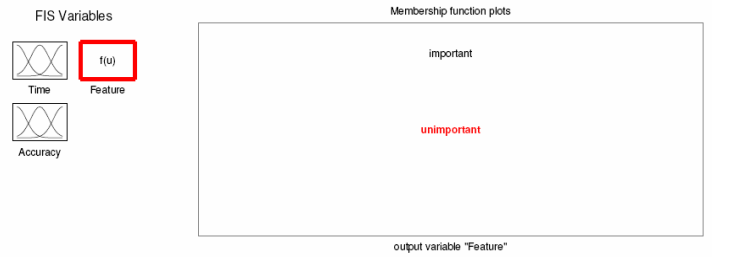
```
[1]  Calculate the Global Accuracy
       Calculate the accuracy and training time of all (41) features
       (Accuracy41, Train41),

       Calculate the accuracy and training time of the features with
       weight >= 1 (Accuracy, Train),

       Pick the Global accuracy
       If Accuracy41 >=  Accuracy
          Global = Accuracy
       Else
          Global = Accuracy41
       End if

[2]  Create the features set
       Sort the features set(S1)in descending order depending on its
        weight values,
       Pick the first three features as an initial features set (S2),
       Calculate the Accuracy and Training time of S2 (Accuracy1,
        Train1)
       If (Global equal or less than Accuracy1)
          Exit;
       Else
          continue_loop= 1,
          count_loop = 0;
          Do while (continue_loop ==1) & (count_loop <= length(S1))
             Add the next feature f(i) from S1 into S2,
              Calculate the accuracy and training time of S2
              (Accuracy2,Train2)
              If (Accuracy2 less than Accuracy1) and (Train2 greater
                   than Train1)
                 Remove f(i) from S2,
                  count_loop = count_loop + 1;
             Else
                Accuracy1 = Accuracy2,
                Train1= Train2,
                count_loop = count_loop + 1;
               If (Global equal or less than Accuracy1)
                 continue_loop = 0,
               End if
            End if
           End while
        End if

[3]  The selected features set = S2
```




Input 1: Time

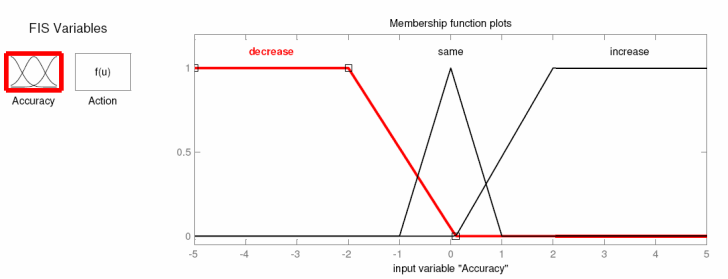



Input 2: Accuracy

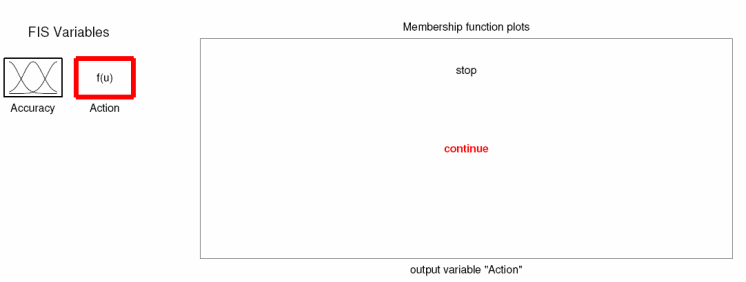



Output: Feature Rank

Figure 3.1 Sugeno Fuzzy Inferencing Model for Local Comparison




Input 1: Accuracy




Output: Action

Figure 3.2 Sugeno Fuzzy Inferencing Model for Global Comparison

*Example (1)*

The following example illustrates the application of the Fuzzy ESVDF algorithm. The algorithm is applied using a DARPA dataset [111] with 6000 samples of which 3000 are normal (50 %) and 3000 are abnormal (50 %), and each instance is characterized by 41 attributes plus a label of either normal or attack.

In the first step of the algorithm, we calculate the global accuracy by taking the minimum of two values: the accuracy of all the features and the accuracy of S1.

The accuracy of all features = 99.70%

The accuracy of the features set (S1) = 99.67%

→ The Global = min (99.70, 99.67) = 99.67%

The second step of the Fuzzy ESVDF algorithm is to build the features set, and this step is done by:

(1) Sorting S1 in descending order by weight value,

S1(29) =[3, 23, 24, 5, 12 ,33 ,34 ,35, 4, 26, 2, 39, 38, 29, 25, 32, 36, 27, 28, 8, 41, 31, 40, 30, 37, 22, 1, 7, 16]

(2) Picking the first three features as an initial features set (S2), and calculating the accuracy and training time of S2 (Accuracy1, Train1)

S2(3) = [3,23,24] → Accuracy1= 96.56%, Train1 = 13.41 sec

Global Comparison: Rule (3) fired → continue adding feature (go to step (3))

(3) Expanding the features set (S2) depends on the defined fuzzy rules

S2(4) = [3,23,24,**5**] → Accuracy2= 98.65%, Train2 = 3.14 sec

Local Comparison: Rule (3) fired → feature (5) is important (keep feature (5))
Accuracy1 ← Accuracy2
Time1 ←Time2

Global Comparison: Rule (3) fired → continue adding feature (add feature (12))

S2(5) = [3,23, 24, 5,**12**] → Accuracy2= 99.20%, Train2 = 3.31 sec

Local Comparison: Rule (6) fired → feature (12) is important (keep feature (12))
Accuracy1 ← Accuracy2
Time1 ← Time2

Global Comparison: Rule (3) fired → continue adding feature (add feature (33))

S2(6) = [3,23, 24, 5,12,**33**] → Accuracy2= 99.37%,Train2=3.78 sec

Local Comparison: Rule (5) fired → feature (33) is important (keep feature (33))
Accuracy1 ← Accuracy2
Time1← Time2

Global Comparison: Rule (3) fired → continue adding feature (add feature (34))

S2(7)=[3,23,24,5,12,33,**34**] → Accuracy2=99.53%,Train2=3.73 sec

Local Comparison: Rule (5) is fired → feature (34) is important (keep feature (34))
Accuracy1 ← Accuracy2
Time1← Time2

Global Comparison: Rule (2) fired → stop adding feature

At the end, the Fuzzy ESVDF is restricted to the features set = S2(7) = [3, 5, 12, 23, 24, 33, 34]

*Example (2)*

In the second example, we apply the proposed approach with the SPECT Heart dataset from UCI Irvine Machine Learning Repository [112]. The dataset contains 267 samples, of which 55 are normal (20.6 %) and 212 are abnormal (79.4 %). Each instance is characterized by 44 attributes plus a label of either normal or abnormal.

In the first step of the algorithm, we calculate the global accuracy by taking the minimum of two values: the accuracy of all the features and the accuracy of S1

The accuracy of all features = 68.97%

The accuracy of the features set (S1) = 70.28%

→ The Global = min (68.97, 70.28) = 68.97%

The second step of the Fuzzy ESVDF algorithm is to build the features set, and this step is done as follows:

(1) Sorting S1 in descending order by weight value,

S1(36) =[43, 40, 42, 37, 14, 1, 13, 4, 34, 31, 44, 27, 30, 8, 10, 2, 32, 16, 22, 15, 26, 6, 21, 7,12, 41, 2, 28, 29, 11, 36, 25, 35, 39, 9, 18]

(2) Picking the first three features as an initial features set (S2), and calculating the accuracy and training time of S2 (Accuracy1, Train1)

S2(3) = [43,40,42] → Accuracy1= 65.28%, Train1 = 0.7 sec

Global Comparison: Rule (3) fired → continue adding feature (go to step (3))

(3) Expanding the features set (S2) depends on the defined fuzzy rules

S2(4) = [43,40,42,**37**] → Accuracy2= 68.33%, Train2 = 0.39 sec

Local Comparison: Rule (6) fired → the feature (37) is important (keep feature (37))
Accuracy1← Accuracy2
Time1 ← Time2

Global Comparison: Rule (3) fired → continue adding feature

S2(5) = [43,40, 42, 37,**14**] → Accuracy2=64.41%,Train2=1.08 sec

Local Comparison: Rule (7) fired → feature (14) is non- important (remove feature (14))

S2(5) = [43,40, 42, 37,**1**] → Accuracy2= 70.97%, Train2 = 0.78 sec

Local Comparison: Rule (6) fired → feature (1) is important (keep feature (1))
Accuracy1← Accuracy2
Time1 ← Time2

Global Comparison: Rule (1) fired → stop adding feature

At the end, the Fuzzy ESVDF is restricted to the features set = S2(5) = [1,37,40,42,43]

### 3.1.3 Experiments and Results

For evaluating the performance of our proposed approach, we choose the Defense Advanced Research Projects Agency (DARPA) KDD-99 benchmark dataset [111]. In addition, we select four smaller datasets from the Irvine Machine Learning Repository (UCI) databases [112]. In this sub-section, we initially describe the contents of the different datasets and the experimental settings, followed by some experimental results and discussion.

#### *Datasets Description*

Five real datasets are considered. The first dataset is KDD-99 data, and the other four datasets are taken from the UCI. The objective is to select a subset for the features using the Fuzzy ESVDF

approach, and then to evaluate these selected features using both Neural Networks (NNs) and Support Vector Machines (SVMs).

*A. The DARPA Dataset*

KDD-99 dataset [111] contains TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN to configure and evaluate Intrusion Detection Systems. It includes three independent sets: whole KDD, 10 % KDD, and corrected KDD. In our experiment, 10 % KDD is used as our training and testing datasets. These datasets contain 24 attack types, which fall into four main classes: *Denial of Service (DoS), Probe, User to Root (U2R),* and *Remote to Local (R2L)*. Both training and testing datasets are made up of a large number of network traffic connections and each data sample is represented with 41 features, plus a label of either normal or attack. Those 41 features can be divided into three groups: the first group includes features describing the commands used in the connections (instead of the commands themselves). These features describe the aspects of the commands that have a key role in defining the attack scenarios (e.g., number of file creations, number of operations on access control files, number of root accesses).

The second group includes features describing the connection specifications. This group includes a set of features that present the technical aspects of the connection (e.g., protocol types, flags, duration, service types, and number of bytes from source).

The third group includes features describing the connection to the same host in the last two seconds (e.g., number of connections having the same destination and using the same service, percentage of connections to the current host that have a reject error, percentage of different services on the current host). In our experiments, we picked two different datasets for training and testing purposes. Each dataset contains 6000 samples; of which 3000 are normal samples (50 %) and 3000 are attack samples (50 %) (i.e., the total number of samples equals 12000).

*B. The UCI Irvine Machine Learning Repository Dataset*

In addition, we test our approach with other four datasets of various sizes. These datasets are selected from the UCI datasets: the SPECT Heart dataset, the WDBC dataset, the Hill and Valley dataset, and the WBC dataset [112].

The first dataset, the SPECT Heart Dataset, describes the diagnosis of cardiac Single Proton Emission Computed Tomography (SPECT) images. It contains 267 samples, of which 55 are normal (20.6 %) and 212 are abnormal (79.4 %). Each instance is characterized by 44 attributes.

The second dataset, Wisconsin Diagnostic Breast Cancer (WDBC), describes characteristics of the cell nuclei present in the image as either benign or malignant. This dataset contains 569 samples, of

which 357 are benign samples (62.74 %) and 212 are malignant samples (37.26 %). Each instance is characterized by 30 real-value attributes.

In the third dataset, the Hill and Valley Dataset, each record represents 100 points on a two-dimensional graph. When plotted in order (from 1 through 100) on the Y co-ordinate, the points will create either a Hill or a Valley. This dataset contains 1212 samples, of which 612 are hill samples (50.5 %) and 600 are Valley samples (49.5 %). Each instance is characterized by 100 real-value attributes.

Finally, the Wisconsin Breast Cancer (WBC) dataset describes characteristics of the cell nuclei present in the image as either benign or malignant. This dataset contains 699 samples;,of which 458 are benign samples (65.52 %) and 241 are malignant samples (34.48 %). Each instance is characterized by nine attributes.

### *Experimental Settings*

To evaluate the performance of our proposed approach, we conducted two experiments. In the first experiment, we chose the DARPA KDD-99 benchmark dataset. In the second one, we picked four different smaller datasets from UCI databases.

*A. The DARPA Dataset*

The dataset used for this experiment is the DARPA KDD-99 dataset, which contains 41 features plus a label of either normal or attack. Through this experiment, we will evaluate our approach [118], [119] by comparing it with the performance of the approaches of [34] and [105] over all 41 features. In [34], they claimed that the best features set includes the six features with largest weight (rank) values. The weight values were evaluated using SVDF (1). In the second approach [105], they applied FS to pick the features set.

Our experiment was split into two main steps. In the first step, we applied the three different approaches (Fuzzy ESVDF [118], [119], the six important features [34], and FS [105]) to select an appropriate features set for the IDS. In the second step, we validated the results by using any classifier type.

In the first step, the proposed algorithms were repeated ten times over the training and testing datasets. Each time about 30 % of the samples were randomly selected as a testing dataset; the remaining 70 % were used as a training dataset of each dataset (we have 12000 samples, and they are split into two datasets, each containing 6000 samples).

NN and SVM classifiers were used to evaluate the proposed algorithms in the second step. We carried out four validation experiments using Fuzzy ESVDF features [118], [119], the six important features [34], FS [105] features, and all 41 features. Each experiment was repeated five times for each dataset (the total number of repetitions for both datasets was ten) and by randomly selecting the training and the testing data using different splitting ratios, which were ((training %) / (testing %): 50/50, 40/60, 60/40, 30/70, and 70/30).

*B. The UCI Irvine Machine Learning Repository Dataset*

Four different datasets were picked from UCI databases for this experiment: SPECT Heart Dataset, WDBC dataset, Hill and Valley dataset, and WBC dataset. Through this experiment, we applied our approach in different domains (each has different number of features) in order to evaluate our approach performance and behaviour with a different number of features. Similar to the DARPA dataset experiments, the experiment was divided into two main steps. First, we applied the proposed algorithm, Fuzzy ESVDF, to select the appropriate features set for each dataset. Second, we validated the results by using SVM and NN.

In the first step, the proposed algorithm is applied ten times with training and testing data. Each time, about 40 % of the samples were randomly selected as the testing dataset; the remaining 60 % were used as the training dataset.

In the same manner as the previous experiment, NN and SVM classifiers were used to evaluate the proposed algorithm in the second step. We carried out four validation experiments: SPECT Heart Dataset, WDBC dataset, Hill and Valley dataset, and WBC dataset. Each experiment was repeated ten times with a random selection of the training and the testing data with different ratios, which were ((training %) / (testing %): 50/50, 40/60, 60/40, 30/70, and 70/30).

*C. Classifiers*

For both experiments, the implementation of the proposed approach used the simple SVM library for SVM [113]. The crossover parameters selection of our SVM included a range of basic SVM parameters, various kernel functions, and their performance arguments. In our experiments, the C parameters could take one of these values: 1, 100, 5000, or 10000. The SVM kernel functions we considered were linear and radial basis kernels. The polynomial kernel was degree 1 and 2, and the coefficient (scale) can be 0.5, 2, 3, or 4. $\sigma$ in a radial basis kernel at either 0.5, 1, 2, or 3.

For evaluation of the different approaches with NN, we used the MTALB BPL toolbox for NN with three layers (an input layer with the number equal to features neurons, a hidden layer with six neurons, and an output layer with one neuron). We used the function “newff” from the MATLAB

toolbox with sigmoidal activation function, performance function "MSE", 45 epochs and a 0.001 learning rate.

### *Experimental Results*

Fuzzy ESVDF was applied to the DARPA KDD-99 dataset and the four different datasets from UCI databases (SPECT Heart Dataset, WDBC dataset, Hill and Valley dataset, and WBC dataset) to select the best features set for the application. In these experiments, we used standard measurements such as *Detection Rate (DR), False Positive Rate (FPR)*, and *overall Classification Rates (CR)* to evaluate the performance of our approach. We defined here *True Positive (TP), True Negative (TN), False Positive (FP)* and *False Negative (FN)* where:

- *True Positive (TP)*: The number of malicious records correctly identified.
- *True Negative (TN)*: The number of legitimate records correctly classified.
- *False Positive (FP)*: The number of records that were incorrectly identified as attacks, though they were in fact legitimate activities.
- *False Negative (FN)*: The number of records that were incorrectly classified as legitimate activities, though they were in fact malicious.

Equations (2) to (4) given as:

$$DR = \frac{TP}{TP + FN} \tag{2}$$

$$FPR = \frac{FP}{TN + FP} \tag{3}$$

$$CR = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

*A. The DARPA Dataset*

Fuzzy ESVDF [118], [119], the six important features [34], and FS [105] approaches were applied to the 41 features to select an appropriate features set for the IDS. To evaluate the approaches, we used SVM and NN classifiers to classify a network traffic record as being either an attack or a normal behaviour. The results of the SVM classifier for Fuzzy ESVDF features, the six important features, FS features, and all 41 features are presented in Table 3.1. Table 3.2 presents the results of the NN classifier for the Fuzzy ESVDF features, the six important features, FS features, and all 41 features. The comparison between the three approaches and using 41 features is done with respect to different performance indicators: number of features, DR, FPR, training time, and testing time.

Table 3.1

Comparison of Fuzzy ESVDF, the six most important features, FS features, and the entire 41 features using SVMs

| **Features Selection Algorithm** | **No. of Features** | **DR (%)** | **FPR (%)** | **Training Time (sec)** | **Testing Time (sec)** |
|---|---|---|---|---|---|
| FuzzyESVDF | 7 | 99.57 | 0.22 | 2.410 | 0.054 |
| 6 important | 6 | 98.20 | 0.39 | 6.008 | 0.114 |
| FS | 8 | 99.23 | 0.35 | 2.246 | 0.056 |
| Non | 41 | 99.62 | 0.32 | 5.182 | 0.170 |

Table 3.2

Comparison of Fuzzy ESVDF, the six most important features, FS features, and the entire 41 features using NNs

| **Features Selection Algorithm** | **No. of Features** | **DR (%)** | **FPR (%)** | **Training Time (sec)** | **Testing Time (sec)** |
|---|---|---|---|---|---|
| FuzzyESVDF | 7 | 99.70 | 0.24 | 221.928 | 0.047 |
| 6 important | 6 | 98.20 | 0.41 | 217.115 | 0.062 |
| FS | 8 | 98.41 | 0.56 | 233.343 | 0.053 |
| Non | 41 | 99.63 | 0.36 | 911.680 | 0.075 |

*B. The UCI Irvine Machine Learning Repository Dataset*

In this experiment, we selected four smaller datasets from UCI databases (SPECT Heart dataset, WDBC dataset, Hill and Valley dataset, and WBC dataset) to test the effectiveness of our feature selection approach (Fuzzy ESVDF) in different domains. We used SVM and NN classifiers to classify a record as being either zero or one (binary classification). The results of the SVM classifier for Fuzzy ESVDF for all datasets are presented in Table 3.3. The results of the NN classifier for Fuzzy ESVDF for all datasets are presented in Table 3.4. The different datasets are compared with respect to different performance indicators: number of features, CR, training time, and testing time. Table 3.5 compares execution times for Fuzzy ESVDF approach for the four different datasets.

Table 3.3

Comparison of different datasets using SVMs

| **Dataset** | **No. Attributes** | **CR (%)** | **Training Time (sec)** | **Testing Time (sec)** |
|---|---|---|---|---|
| SPECT Heart | Selected Set (5) | 76.73 | 0.330 | 0.010 |
| | Complete Set (44) | 69.43 | 0.376 | 0.015 |
| WDBC | Selected Set (4) | 96.65 | 0.452 | 0.000 |
| | Complete Set (30) | 96.42 | 0.476 | 0.012 |
| Hill and Valley | Selected Set (11) | 69.13 | 23.588 | 0.088 |
| | Complete Set(100) | 66.77 | 42.450 | 0.152 |
| WBC | Selected Set (3) | 96.75 | 0.678 | 0.012 |
| | Complete Set (9) | 94.72 | 0.850 | 0.016 |

Table 3.4

Comparison of different datasets using NNs

| **Dataset** | **No. Attributes** | **CR (%)** | **Training Time (sec)** | **Testing Time (sec)** |
|---|---|---|---|---|
| SPECT Heart | Selected Set (5) | 74.14 | 33.940 | 0.006 |
| | Complete Set (44) | 66.77 | 325.534 | 0.030 |
| WDBC | Selected Set (4) | 94.85 | 38.031 | 0.012 |
| | Complete Set (30) | 94.63 | 206.144 | 0.013 |
| Hill and Valley | Selected Set (11) | 77.93 | 105.638 | 0.019 |
| | Complete Set(100) | 75.84 | 2488.83 | 0.047 |
| WBC | Selected Set (3) | 95.91 | 43.738 | 0.016 |
| | Complete Set (9) | 95.05 | 72.475 | 0.015 |

Table 3.5

Execution time comparison for the different datasets

| **Dataset** | **No. of All Features** | **No. of Selected Features** | **Execution Time (sec)** |
|---|---|---|---|
| SPECT Heart | 44 | 5 | 125.093 |
| WDBC | 30 | 4 | 87.215 |
| Hill and Valley | 100 | 11 | 3000.025 |
| WBC | 9 | 3 | 0.725 |

### *Discussion*

As shown in Table 3.1 and Table 3.2, a comparison of our approach (Fuzzy ESVDF) against all 41 features reveals a dramatic reduction in model building time with the reduced features using Fuzzy ESVDF, as the proposed features selection algorithms have cut 83 % of the total number of features (Fuzzy ESVDF selects seven features from among the 41 features). When an SVM classifier is used, the DR and FPR for our approach and the entire 41 features are nearly the same. However, for the training and testing time, the results show a significant improvement with our approach. The training and testing times decrease by more than fifty per cent, as opposed to when all 41 features are used. Also, evaluating the proposed approach with NN, DR and FPR do not show much difference to that of using all 41 features. But training and testing times show an obvious improvement: it cuts 75.66 % from the required training time for all features, and 69.41 % from the required testing time for all features, which means that the proposed algorithms can achieve high accuracy with less training and testing time. Moreover, the experimental results show that SVM outperform NN in classification accuracy and training time.

Comparing our approach with that of Mukkamala *et al.* [34], we see that they extracted six important features as the features selection set. Table 3.1 and Table 3.2 show that by using NNs or SVMs, Fuzzy ESVDF is better than using the six important features in terms of classification performance (DR and FPR). For the SVM classifier, our approach outperforms using the six important features in DR (DR increases from 98.20 % to 99.57 %), FPR (FPR reduces from 0.39 % to 0.22 %), training time is cut 60 % from the required training time for the entire 41 features, and 62.5 % from the required testing time. For the NN classifier case, there is an improvement in DR (DR increases from 98.20 % to 99.70 %) and FPR (FPR reduces from 0.41 % to 0.24 %); however, the training and testing are nearly the same. In general, Fuzzy ESVDF results are better than the six important features approach because limiting the number of selection features to a specific value (e.g., 6 or 11) as an indicator for highest rank value may affect the system performance. So, we need a process that uses this rank value (weight value) to select an appropriate feature subset. Therefore, SVDF needs to be manipulated to recover from these limitations: in our case, this is accomplished by applying the FS algorithm.

Comparing Fuzzy ESVDF against FS approaches [105], Table 3.1 shows that both approaches have nearly the same performance. However, Table 3.2 shows an obvious improvement in terms of DR and FPR (DR increases from 98.41 % to 99.70 %, and FPR decreases from 0.56 % to 0.24 %). The training and testing times are nearly the same in both approaches.

In summary, for the first experiment, the proposed approach for features selection gives an excellent performance in terms of training and testing times while retaining high classification accuracy, regardless of the classifier used. Fast training and testing help to build lightweight IDS; they also facilitate the retention or modification of the system and allow for the use of this model in a real-time intrusion detection environment. Moreover, with the reduction of the features number, we can identify relevant attack-specific features, simplifying the study and analysis of the behaviour of each of these attacks. On the other hand, our approach does not guarantee the selection of the optimal features set. In all cases, however, it shows a dramatic improvement in the detection process.

For the second experiment, evaluating our approach with four different smaller datasets from the UCI databases, Table 3.3 and Table 3.4 show that there is a dramatic reduction in the number of features for all datasets after the application of Fuzzy ESVDF. For the SPECT Heart dataset, the number of features is reduced from 44 to 5 (it eliminates nearly 88.6 %). For the WDBC dataset, the number of features is reduced from 30 to 4 (it cuts nearly 86.7 %). For the Hill and Valley dataset, the number of features is reduced from 100 to 11 (it cuts nearly 89 %). Finally, for the WBC dataset, the number of features is reduced from 9 to 3 (it cuts nearly 66.67 %).

For the SPECT Heart dataset, by using SVM, the CR based on the selected features set is 76.73 %, which is better than the CR for the complete features set, which is 69.43 %. The training time and testing time are nearly the same in both cases, as shown in Table 3.1. However, the NN classifier shows an obvious improvement in both training and testing times. The five selected features from the Fuzzy ESVDF approach cut 89.57 % from the required training time for all features, and the testing time is cut by 78.67 %. The CR is also improved, as shown in Table 3.2.

For the WDBC dataset, by using SVM, the CR, training time, and testing time based on the selected features set are very near to their values in the system which used the entire features, as shown in Table 3.3. Table 3.4 shows a significant improvement in training time for the selected features. The proposed algorithm cuts 81.55 % from the required training time for all features. However, CR and testing time in both experiments (using the four features selected from the Fuzzy ESVDF algorithm, and using the entire 30 features) are nearly the same.

For the Hill and Valley dataset, Table 3.3 shows a significant improvement in CR, training time, and testing time for the selected 11 features (attributes) from the complete 100 features set. On the other hand, Table 3.4 shows that the CR for both features sets (11 selected features and the entire 100 features) are nearly the same. However, there is an obvious improvement in training time (it is

reduced by 95.76 % as compared with the required training time for all features), and testing time (it is reduced by 59.57 % as compared with the required testing time for all features).

For the WBC dataset, by using SVM, the overall system performance is improved based on the selected features set, as shown in Table 3.3. On the other hand, Table 3.4 shows significant improvement in training time (is reduced by 39.65 % as compared with the required training time for all features) based on the selected features. However, the CR and testing time are nearly the same in both cases (selected and all features).

Comparing the different datasets' execution time, Table 3.5 shows that the proposed approach, Fuzzy ESVDF, becomes slow when the number of features increases to 100. When the number of features is around 50 (in the case of SPECT Heart dataset), the algorithm is reasonably fast, but when this number doubles (to 100 features), execution time increases greatly. Also, in the case of the WBC dataset, the number of features is 9, but when the number of features triples (in the case of the WDBC dataset), the execution time more than triples. On the whole, this amount of time does not depend on the number of features alone. It does depend, however on how fast SVM are, because the ranking approach depends on the system performance (CR and training time) that is calculated by SVM. Moreover, the SVDF used in this approach also depends on SVM.

In summary, the experimental results demonstrate the feasibility of the proposed approach. The proposed approach, Fuzzy ESVDF, for a features selection based on SVDF and FS with the fuzzy inferencing model, gives the best performance in terms of training and testing times, while retaining high classification accuracy regardless of the classifier used. Consequently, the selected features subset is representative and informative and, thus, can be used to replace the complete features. In addition, this approach is considered to be a features selection approach regardless of the type of classifier used, making this approach a suitable features selection method for many applications. Lastly, the proposed algorithm is simple and does not require that many parameters be initialized, and further, does not need heavy computational resources. This facilitates the retention or modification of the system design and allows this model to be used in a real-time environment.

### 3.1.4 Summary

SVDF is used to rank the input features by giving a weight value to each of them. Using the weights alone, however, as proposed in previous works [25], [26], [34], [114], we are unable to specify the appropriate features set for a detection process because selecting features with the highest rank values (weight) cannot guarantee that combining these features can create the best features set based on the

correlations among candidate features. Moreover, limiting the number of selection features to a specific value (e.g., 6 or 11 as mentioned in the previous works) as an indicator for highest rank value may affect system performance.

Our new approach overcomes these limitations by proposing a Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF) [118], [119]. The Fuzzy ESVD approach improves the classification process by integrating the feature ranking technique (evaluated by SVDF) with the features selecting technique (applied by FS). The fuzzy inferencing model is used to accommodate the learning approximation and the small differences in the decision-making steps of the FS approach. The proposed approach (Fuzzy ESVDF) has a wealth of advantages that make it attractive for many features selection applications.

First, by employing a reduced number of features SVM may be more advantageous than other conventional features selection methods [13], [22], [23], [65]. The advantage is conspicuous for many applications, as our experiments show. With SVM, a satisfactory performance can be obtained much more easily than with other approaches.

Second, by evaluating the features weights through SVDF and then selecting between these features through the application of the FS algorithm, this approach is able to efficiently select the appropriate features set for the classification process.

Third, ESVDF is considered to be a features selection approach regardless of the type of classifier used, making this approach a suitable features selection method for many applications, as we showed through our experiments.

Finally, this approach is simple and efficient, and it does not require parameters initialization, which facilitates modification and enhancement.

To evaluate the proposed approach, we used SVM and NN classifiers and a KDD-99 dataset for our experiments. The experimental results demonstrate that our approach can reduce training and testing times while retaining a high classification accuracy for IDS. In addition, we used four different datasets from UCI Irvine Machine Learning Repository (SPECT Heart Dataset, WDBC dataset, Hill and Valley dataset, and WBC dataset) to test the effectiveness of our features selection approach in different domains. The experimental results demonstrate that our approach can reduce the training and testing times with high classification accuracy for any application in general. Thus, it combines good effectiveness with high efficiency. It produces an efficient features subset, so it provides an

effective solution to the dimensionality reduction problem in general. On the other hand, the efficiency of Fuzzy ESVDF depends on how SVM are able to classify the dataset, which may obstruct the modification and maintenance processes and impede the use of this approach in some types of applications. Moreover, it can not guarantee the optimal solution in terms of minimizing the number of features, but in all situations it gives a considerably reduced number of features with excellent performance results.

## 3.2 IDS Classification Scheme

Previous studies [19], [115], [116], [117] showed that desired features for the IDS depend on the type of attack. Accordingly, as each TCP/IP network layer is vulnerable to a specific type of network attack, each TCP/IP network layer needs a specific type of IDS. In this section, we propose a new classification scheme for IDS depending on the TCP/IP network model [152]: Application layer IDS (AIDS), Transport layer IDS (TIDS), Network layer IDS (NIDS), and Link layer IDS (LIDS). This new scheme can improve the overall performance of IDS for the following reasons. First, each of these IDS types is specialized to a specific network device. So, the detection process will be distributed among all TCP/IP network model layers through the network devices.

Moreover, as is known, firewalls operate at different TCP/IP network layers by using different criteria to restrict traffic, but this step is far from running an entirely secure network as not all traffic will go through a firewall. They can protect the network from attackers coming from outside the network, but they cannot protect it from attackers coming from inside the network; therefore, an IDS must be allocated as a second line of defense behind the firewalls. In addition, the attacks usually gain access to the network through the network devices distributed through different TCP/IP network layers as entry points, and in order to adequately address security, all possible avenues of entry must be evaluated and secured. An IDS must therefore be allocated to these entries or network devices.

Finally, splitting the detection process into different levels and stages reduces the computational load on the system and improves its scalability and performance. Accordingly, categorizing IDS into different types depending on the TCP/IP layers becomes an essential issue for improving the overall system detection ability.

### 3.2.1 TCP/IP Model and Attack Classification

A computer network is simply a system of interconnected computers using different network models. There are two famous standard network models: the ISO/OSI model and the TCP/IP model, which are

based on a layered concept. The layered concept of networking was developed to accommodate changes in technology. Each layer of a specific network model may be responsible for different functions of the network. Each layer passes information up and down to the next subsequent layer as data is processed. The TCP/IP network model defines a set of rules to enable computers to communicate over the network, specifying how data should be packaged, addressed, shipped, routed, and delivered to the right destination. The TCP/IP family uses four layers while ISO/OSI uses seven layers, as shown in Figure 3.3. The TCP/IP and ISO/OSI systems differ from each other significantly, although they are very similar on the network and transport layers.

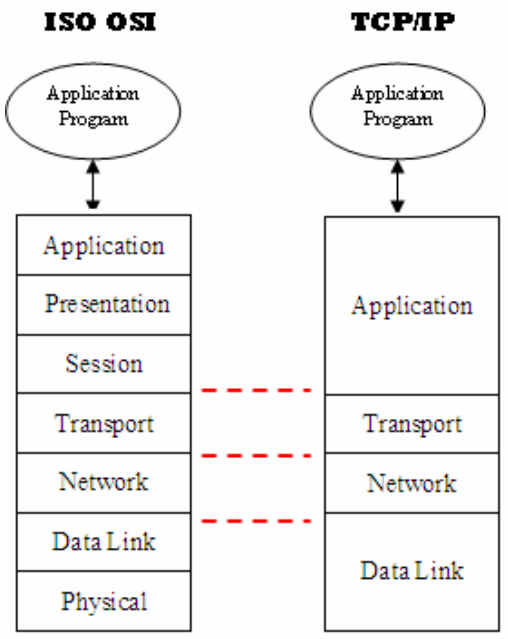



Figure 3.3 Comparison of TCP/IP and ISO OSI network models

In the TCP/IP model, each layer has its own functionality and services. The Data Link layer defines physical media and cables. The Network layer handles the end-to-end communications; it is used for basic communication, addressing and routing. The Transport layer is responsible for end-to-end message transfer capabilities independent of the underlying network, along with error control, fragmentation, and flow control. At the end, the Application layer provides network services to end-users such as Web browsing (HTTP), remote access (Telnet), File transfer (FTP), and other services.

So, each layer of communication has its own software, hardware, configuration, protocols, and usage. Accordingly, each layer will have its own unique attacks and security challenges, which means each layer needs a specific protection process.

The Data Link layer defines the device driver and network hardware (network interface card). It is responsible for node-to-node (hop-to-hop) frame delivery between the Internet layer interfaces of two different hosts on the same link. This layer describes the protocols used to describe the local network topology and the interfaces needed to effect transmission of Internet layer datagrams to next neighbor hosts such as SLIP (Serial Line Internet Protocol), CSLIP (Compressed SLIP), PPP (Point to Point Protocol), Ethernet, Token Ring, Frame Relay, ATM, etc. Accordingly, this layer is vulnerable to MAC Attacks, DHCP (Dynamic Host Configuration Protocol) Attacks, ARP (Address Resolution Protocol) Attacks, STP, and VLAN-Related Attacks. The Data Link layer can be a very weak link in terms of security, and worse yet, it can affect the upper layers by causing service disruptions or security breaches.

The Network layer of the TCP/IP model provides end-to-end packet delivery. It handles basic communication, addressing and routing, and manages the movement of packets around the network; it defines the addressing and routing structures used for the TCP/IP protocol suite. The primary protocols in this scope are IP (Internet Protocol), ICMP (Internet Control Massage Protocol), ARP (Address Resolution Protocol), RIP (Routing Information Protocol), and RARP (Reverse Address Resolution Protocol). Typically, in this layer, the attackers exploit the fact that IP does not have a robust mechanism for authentication such as IP Spoofing, IP Session Hijacking, Ping to Depth, and Source Routing,

The Transport layer handles end-to-end message transfer capabilities independent of the underlying network, while providing reliable delivery. It handles the flow of data among applications, and segments data into packets for transport over the network. This is where flow-control, error-correction, and connection protocols exist, such as TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and ICMP (Internet Control Message Protocol). This layer is especially vulnerable to Denial of Service (DoS) attack (or Distributed Denial of Service (DDOS) attack), SYNC Flood, UDP Bomb, and Port Scan.

Finally, the Application layer provides network services such as browsing, e-mail, file transfer, remote access, etc. Accordingly, the common protocols used are HTTP (HyperText Transfer Protocol), DNS (Domain Name System Protocol), FTP (File Transfer Protocol), IRCP (Internet Relay Chat Protocol), and POP/ POP3 (Post Office Protocol). In order for the attackers to exploit system

vulnerabilities more effectively, they have developed several sophisticated ways to attack the application layer, such as Buffer overflow, Trojans, Backdoor, Cross Site Scripting, etc. The most popular application attack types are E-Mail Attacks, FTP Attacks, Web Attacks, and DNS Attacks.

On the whole, each layer of the TCP/IP layer has its own software, hardware, configuration, protocols, and usage. Each layer needs a specific protection process against each type of attack. Therefore, network security should be addressed at each TCP/IP layer for different vulnerabilities, security challenges, and attack types. Table 3.6 presents common attacks for each TCP/IP layer.

Table 3.6

The Common Attacks for each TCP/IP layer

| TCP/IP Layer | Common Attacks |
|---|---|
| Application Layer | Java, ActiveX, and Script Execution<br>WinNuke<br>E-Mail Attacks<br>FTP Attacks<br>Web Attacks<br>DNS Attacks |
| Transport Layer | SYN Flood<br>UDP Bomb<br>Port Scan<br>Landc<br>TCP Port Scans<br>UDP Application Attacks<br>RIP Attacks |
| Network Layer | Ping Flood<br>Ping of Death<br>IP Spoof<br>Address Scanning<br>Source Routing<br>ICMP Attacks |
| Data Link Layer | Sniffer/ Decoding<br>MAC Address Spoofing<br>WEP Attacks |

### 3.2.2 TCP/IP Attack Classification and IDS Categorization

Network security should be addressed at each TCP/IP network layer for different vulnerabilities and attack types. Features that tend to detect a particular type of attack may not be useful in the detection of other attack types. Many researchers [19], [115], [116], [117] have shown that the choice of network features for IDS is dependent on the network attack type to be detected. Some features were good for detecting network attack traffic patterns, while other features were good for detecting

transport attack traffic patterns. Hence, studying the nature of the IDS environment is an important issue for choosing the appropriate features to analyze the traffic pattern. Moreover, the features selection phase in IDS implementation has a large impact on performance. Reducing the number of features can improve system performance, speed up execution and training times, allow the extraction of easily interpretable rules, and reduce the measurements and storage requirements (explained in Section 3.1). Based on the attack classification (Data Link layer attacks, Network layer attacks, Transport layer attack, and Application layer attacks) we will select the appropriate features for each TCP/IP layer, so that for each layer there will be a specialized IDS. It is important that the different types of security attacks be recognized in order to select the appropriate countermeasures.

Furthermore, firewalls are a crucial piece of the network security, but this step is far from creating an entirely secure network, because of their limitations. The firewalls can protect the network from attackers that come from outside the network (intrusions), but they cannot protect it from attackers that come from inside networks (misuses). Also, they can only guard against the traffic that passes through them; they have only minor control over the data that passes through them. Certain traffic types, such as a remote user's dial-up connection to a Remote Access Server (RAS), would bypass the firewall entirely. In addition to other limitations (such as an inability to tell the user that it has been incorrectly configured), a firewall can't notify the administrator if someone has hacked into the network. Proper configuration is a must to maintain the efficacy of any firewall system, and they should be updated periodically to ensure that they are current with the internal and external environment of the network. Activity logs should also be checked on a regular basis to find attempted and successful intrusions. Accordingly, the IDS must be allocated as a second line of defense behind the firewalls, and as the firewalls operate at different TCP/IP network layers, the IDS also needs to be allocated in same manner as the firewall.

Moreover, the IDS should monitor traffic at entry points (network devices) on the network or interconnected set of networks which are considered to be the target of the intruder. By securing these devices, which are distributed through the network layers, the overall security of the system will be improved, the detection process can be performed at any point where enough information is available, and the data can be collected from multiple sources. This combines the best characteristics of traditional Router-based, Network-based, and Host-based IDS.

Finally, splitting the detection process into different levels (layers) reduces the computation load on the system and improves its scalability and performance, as the experiments in the next section will

show. Accordingly, categorizing the IDS into different types depends on the TCP/IP network model becomes an essential issue for improving overall system detection ability and scalability.

Most of the work done [19], [20], [28], [105] falls into the realm of splitting detection into different attack types. These are broadly categorized into four groups: Probes, Denial of Service (DoS), User to Root (U2R), and Root to Local (R2L). They are then designed a specific IDS for each of these attack categories. This categorization does not participate in reducing the system load, or improving its scalability. Others [10], [22] concentrated on analyzing and detecting each attack separately, which is considered to be impractical with the fast pace of change in attack tools. In this chapter, we propose a new IDS classification scheme that depends on the TCP/IP network model [152], which distributes the detection process between different network devices, thus improving both system performance and scalability, as the experiments will show in Section 4.2.3.

### 3.2.3 IDS Classification Scheme based on TCP/IP

We propose four different types of IDS based on the TCP/IP network model [152] - AIDS, TIDS, NIDS, and LIDS - to accommodate detecting different attack types in each TCP/IP network layer and to improve overall IDS performance, efficiency, and scalability. Figure 3.4 shows the architecture for the proposed approach. The proposed approach categorizes the IDS into four types, with each type responsible for the different network devices that are distributed through the TCP/IP network layers. This would include routers in the Network layer, switches in Data Link layer, etc..

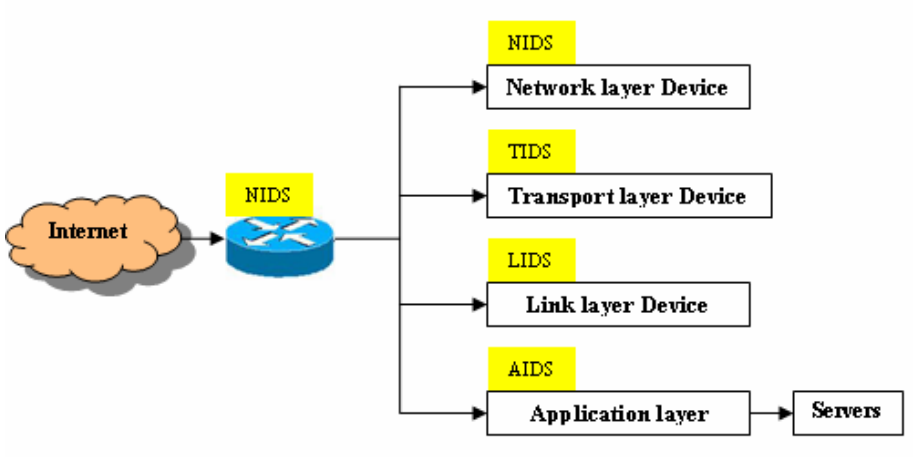



Figure 3.4 System Architecture

The NIDS operates on the Network layer devices (i.e. router) that in turn are located at the network border or on an isolated host connected to the network layer devices, allowing it to analyze the traffic that passes between different networks as it shown in Figure 3.5. For the Transport layer devices, the TIDS is loaded on either Transport layer devices (i.e. switch) or on an isolated machine that is connected to the transport layer devices, allowing it to analyze network traffic that enters the subnet (Figure 3.6). The LIDS is also installed on either the Link layer devices or on an isolated machine that is connected to the transport layer devices to detect different transport attacks. Finally, AIDS refers to the class of intrusion detection systems that reside on and monitor an individual host machine. The AIDS must be loaded on each workstation in the network as it is shown in Figure 3.8. Each IDS type has its own features set depending on its TCP/IP network layer; so instead of using all connection features as has been the case in previous works, we use only a specific number of features for each IDS. The framework architecture of the integrated IDSs is shown in Figure 3.8.

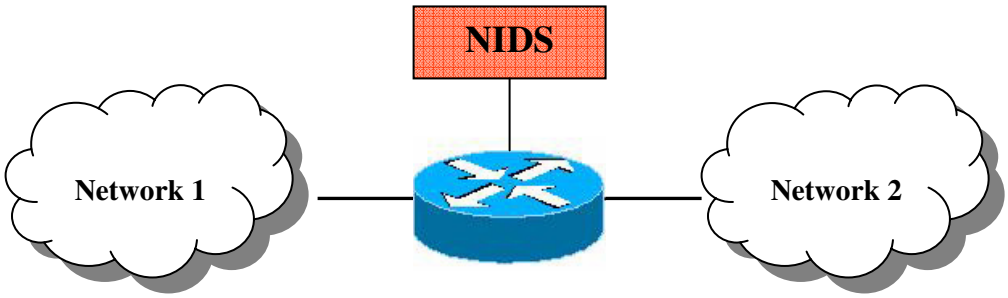



Figure 3.5 The Network IDS (NIDS)

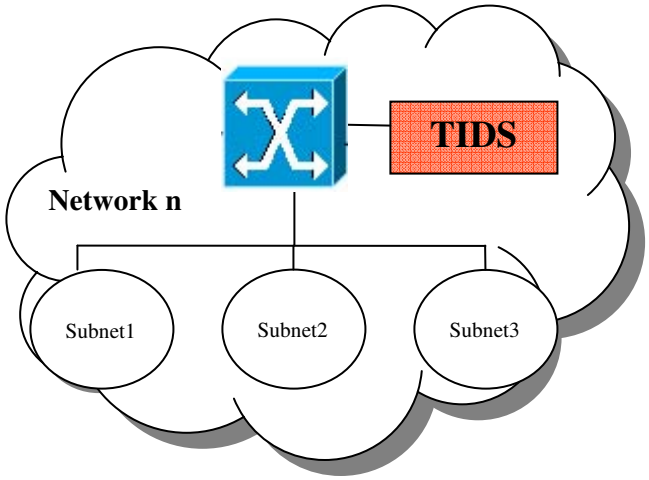



Figure 3.6 The Transport IDS (TIDS)

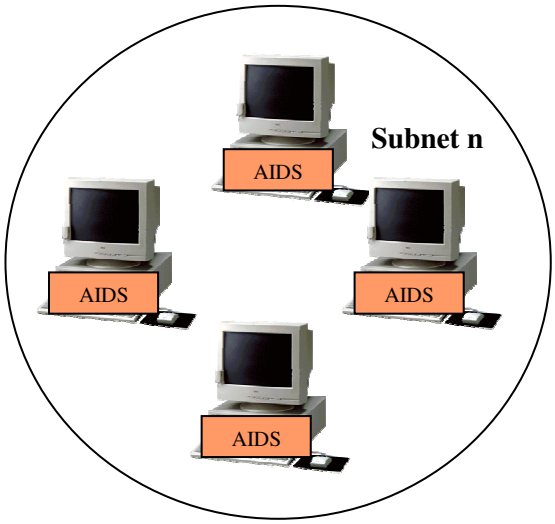



Figure 3.7 The Application IDS (AIDS)

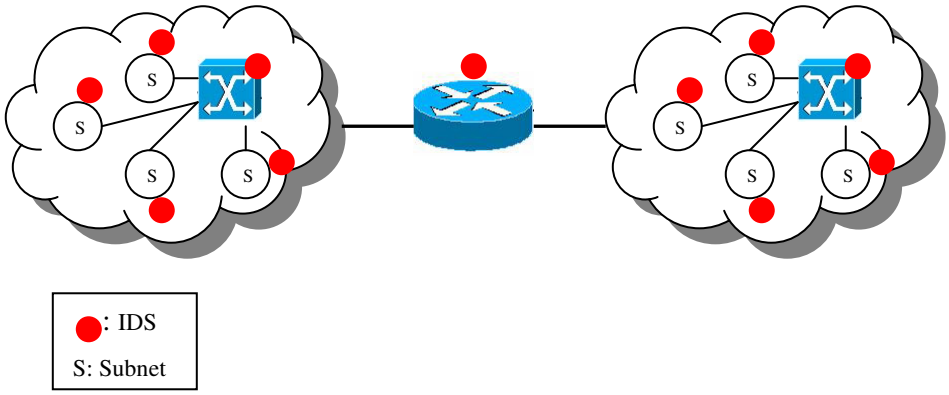



Figure 3.8 The Framework Architecture

To design the different types of IDS (AIDS, TIDS, NIDS, and LIDS), we can use any features selection approach to pick the appropriate features set for each layer in the TCP/IP model. In our case, we will use the Fuzzy ESVDF approach that it is explained in Section 3.1 [118], [119] to select the appropriate features for each IDS type (the LIDS is not covered in this study because there are insufficient link layer attack samples).

To apply the Fuzzy ESVDF approach, there are two main steps. First, we prepare different datasets for training and testing purposes. Each IDS type has its own dataset; for example the AIDS dataset contains normal behaviour and application layer attack samples, and the NIDS dataset contains

normal behaviour and network layer attack samples. In the second step, we apply the Fuzzy ESVDF approach for each dataset to select the most effective features set for each IDS type.

The Fuzzy ESVDF [118], [119], as it is described in Algorithm 3.1, is based on a Support Vector Decision Function (SVDF) (1) and Forward Selection (FS) approach with a fuzzy inferencing model to select the best features as inputs for an IDS. The algorithm is iterative, where each iteration consists of two steps: feature ranking and feature selecting. First, feature ranking, is evaluated by SVDF to rank each specified candidate feature. Then feature selecting (FS) is applied, with the fuzzy inferencing model, to select the features according to a set of rules based on a comparison of performance.

The experimental results in the next section will show that each IDS type has its own features and that is because each TCP/IP network layer is subject to its own attacks. Therefore, classifying IDS depending on the network layers is an essential issue to improve system accuracy, scalability, generality, and to speed up the detection process for the IDS.

### 3.2.4 Experiments and Results

For evaluating the effects of categorizing the IDS into different types depends on the TCP/IP network layers (AIDS, TIDS, NIDS, and LIDS) in improving both system performance and scalability, we choose the DARPA KDD-99 benchmark dataset [111]. In this sub-section, we initially describe the contents of the used dataset. Then, the experimental settings are presenting; followed by some experimental results and discussions.

#### *Datasets Description*

We used the same dataset (KDD-99 dataset [111]) as was used in previous section (Section 3.3.1). In these experiments, we picked two different datasets for training and testing purposes. Each dataset contained 6000 samples, of which 3000 were normal samples (50 %) and 3000 were attack samples (50 %) (i.e., the total number of samples was 12000).

#### *Experimental Settings*

Our experiment was split into three main steps. In the first step, we prepared the different datasets for AIDS, TIDS, NIDS, and all layers IDS (the LIDS was not covered in this study). Second, we applied the features selection approach (Fuzzy ESVDF) to each dataset in order to select the most effective features set for each of IDS type. Finally, we evaluated the features set results using Neural Networks (NN) and Support Vector Machines (SVM) as two different classifiers.

In the first step of the experiment, we prepared the four datasets for each IDS type. For AIDS, the dataset we used contained the normal behaviour patterns and the application layer attack patterns, such as: back, pod, smurf, buffer_overflow, loadmodule,perl, guess_passwd, imap, multihop, warezmaster, ftp_write, nmap, and satan. For the TIDS, the dataset contained the normal behaviour pattern and transport layer attack patterns: land, neptune, teardrop, buffer overflow, port sweep, and nmap. The NIDS dataset contained only smurf, pod, overflow, and IP sweep as the network layer attacks, in addition to the normal behaviour patterns. Finally, all layers of the IDS dataset contained all attack types and normal behaviour patterns.

For the second step, applying the features selection approach, we used the Fuzzy ESVDF [118], [119] approach to select the most effective features for the three IDS types (AIDS, TIDS, and NIDS) and all layers of IDS. The proposed algorithms were performed ten times for each IDS type over training and testing data. Each time about 30 % of the samples were randomly selected as the test data; the remaining 70 % were used as the training data (we had 12000 samples, which were split into two datasets each containing 6000 samples).

Finally, the evaluation was done using NN and SVM classifiers for AIDS, TIDS, NIDS and all layers of IDS features. Each experiment was repeated five times for each dataset (the total number of repetition for both datasets was ten) and by randomly selecting the training and the testing data using different splitting ratios which were ((training %) / (testing %): 50/50, 40/60, 60/40, 30/70, and 70/30).

### *Experimental Results*

We applied the Fuzzy ESVDF approach [118], [119] on 41 features to select the best features set for each type of IDS (AIDS, TIDS, NIDS, and all layers). To evaluate our approach, we used NN and SVM classifiers. They classify a network traffic pattern as being either an attack or a normal behaviour. The results of the classifiers performance for AIDS, TIDS, NIDS and all layers by using the Fuzzy ESVDF approach are presented in Table 3.7. The comparison between the different IDS types was done with respect to different performance indicators: number of features, CR, training time and testing time. On the other hand, Table 3.8 shows the results after swapping the features set between AIDS, TIDS, and NIDS by using the Fuzzy ESVDF approach and the evaluation was done using an NN classifier. The selected features from the Fuzzy ESVDF for each layer are listed in Table 3.9.

Table 3.7

Comparison between All layers, Application, Transport, and Network layer for Fuzzy ESVDF Approach

| Classifier Method | IDS Type | Number of Features | CR (%) | Training Time (sec) | Testing Time (sec) |
|---|---|---|---|---|---|
| Support Vector Machine (SVM) | All Layers IDS | 7 | 99.34 | 4.834 | 0.030 |
| | AIDS | 5 | 99.62 | 2.594 | 0.022 |
| | TIDS | 4 | 99.75 | 1.586 | 0.020 |
| | NIDS | 4 | 99.73 | 2.328 | 0.020 |
| Neural Network (NN) | All Layers IDS | 7 | 99.41 | 180.650 | 0.038 |
| | AIDS | 5 | 99.73 | 162.734 | 0.034 |
| | TIDS | 4 | 99.84 | 139.296 | 0.031 |
| | NIDS | 4 | 99.77 | 144.281 | 0.032 |

Table 3.8

Swapping features between IDS types using Fuzzy ESVDF and evaluated it by NN

| Detection Layer | Used Features | CR (%) | Training Time (sec) | Testing Time (sec) |
|---|---|---|---|---|
| Application | Transport | 85.04 | 139.750 | 0.041 |
| Application | Network | 98.43 | 141.219 | 0.041 |
| Application | Application | 99.73 | 162.734 | 0.034 |
| Transport | Application | 98.98 | 162.172 | 0.031 |
| Transport | Network | 98.78 | 149.641 | 0.031 |
| Transport | Transport | 99.84 | 139.296 | 0.031 |
| Network | Application | 98.61 | 163.828 | 0.035 |
| Network | Transport | 91.16 | 148.406 | 0.038 |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |

Table 3.9

The features set for different IDS types by using Fuzzy ESVDF approach

| IDS Type | Feature index |
|---|---|
| NIDS | 3, 12, 37, 40 |
| TIDS | 2, 3, 12, 32 |
| AIDS | 3, 5, 12, 31, 36 |
| IDS (all layers) | 3, 4, 5, 12, 23, 24, 33 |

### *Discussion*

By using the Fuzzy ESVDF approach, as shown in Table 3.7, splitting the detection process into different layers improves both the system performance and the scalability comparing with all IDS layers. Improving the performance is in terms of increasing the classification accuracy to more than 0.28 % with the SVM classifier and 0.32 % with the NN classifier. Also, the training time and the testing time have been decreased for both SVMs and NNs. Improving the scalability is accomplished by reducing the features number. It is reduced from seven features to five features in AIDS, four features in TIDS, and four features in NIDS, as shown in Table 3.9.

Each layer in the TCP/IP network mode is subject to a specific form of attack, and therefore needs a custom IDS to face those attacks. Table 3.8 shows system performance in a case where the features are swapped between the three different layers (AIDS, TIDS, and NIDS) using the Fuzzy ESVDF approach. As shown, if we use the application features in the transport layer the accuracy will drop to 85.04 % from 99.73 % in application layer. The training and testing time do not show much change because they depend more on the number of features. Also, when the application features are used in the network layer, system accuracy will be affected, dropping to 98.43 % from 99.73 % in the application layer. The same situation is seen in the two other layers.

So, the choice of features depends on the network attack type to be detected: when we swap the features between the layers, the system performance will be affected. Some features were good for detecting application attacks, other features were good for network attacks. Consequently, studying the nature of the IDS environment and the behaviour of the attacks are important issues for choosing the appropriate features to analyze the traffic pattern.

As shown in Table 3.9, the features selected for detecting intrusion at all layers do not cover all the features for the separate layers, which means that when we are more specific on the types of attack,

the features become more accurate because each attack has its own behaviours depending the nature of the layer that it tried to attack. Moreover, each IDS type has its own custom attacks and therefore needs its own custom protection. Therefore, each attack type has its own behaviour and design, which leads us to analyze each attack-type pattern with its own features. However, there is some overlapping between the IDS features and that comes from the overlapping between some attacks. Some attacks target more than one layer such as "nmap", which attacks transport and application layers. Others can attack all layers such as the "buffer overflow" attack.

### 3.2.5 Summary

In general, security can take three main forms: (1) end-to-end security at the TCP/IP application layer, (2) end-to-end security at the TCP/IP transport layer, and (3) link-to-link security at the TCP/IP network and link layers. In this section, we propose a new classification scheme for IDS depending on the TCP/IP network model that accommodates the three main forms of security measures [152]. This classification scheme improves the performance and scalability of the IDS. The performance improvement is in terms of improving the system detection ability and the time performance, which is accomplished through three main points. First, each IDS type can be specialized to detect a specific category of attack depending on the layer. For example, to place IDS in the router, we need to use NIDS, which knows much about router attack behaviour. Secondly, by distributing the IDS through the TCP/IP layers as the second level of defense after the firewall, the firewall will be supported by the IDS and the overall system security will be improved. Furthermore, it is known that one of the major issues in network security is to secure network devices, which are represented as system entry points for the attacks. Also, by this approach, we can integrate intrusion-related information distributed around the network. Hence, by designing a specialized IDS for each one of network layer, the overall system performance will be improved. The proposed approach can also improve system scalability in terms of reducing the number of needed features (five features in AIDS, four features in TIDS, and four features in NIDS). Therefore, splitting the IDS into sub-systems can accommodate reduced system scalability and improve its performance.

We have implemented the different IDS types by using the Fuzzy ESVDF approach to select the appropriate features set for each IDS type, and validated their performance by using NN and SVM classifiers. Experimental results demonstrate that our approach improves system accuracy, efficiency (training time and testing time), and scalability.

## 3.3 Conclusion

Building a lightweight IDS is an important issue in intrusion detection, when considering how to improve IDS efficiency, performance, and scalability. In addition, a lightweight IDS is flexible enough to be used as permanent elements of the network security infrastructure, and is easily incorporated into any network security architecture with minimal disruption to operations. In this chapter, we built a lightweight IDS by applying two different approaches. In the first approach, the features selection approach, we used a novel features selection algorithm based on a Support Decision Function (SVDF) and Forward Selection (FS) approach with a fuzzy inferencing model called Fuzzy ESVDF [118], [119]. The Fuzzy ESVDF is able to significantly decrease training and testing times while retaining high detection rates. In addition, it is simple and efficient, and it does not require parameters initialization, which facilitates a modification and enhancement process.

The second approach employs a new IDS classification scheme. This scheme classifies the IDS into different categories based on the TCP/IP network model (AIDS,TIDS, NIDS, and LIDS) [152]. By designing a specialized IDS for each layer in the TCP/IP network model, overall system performance will be improved, as will the system scalability, generality, and extensibility. In addition, the new classification scheme can reduce intrusion influences and damage that may occur as a result of detection attacks in the first stage (higher or lower TCP/IP layer) before they can enter the network.

We have implemented a number of experiments to evaluate the first approach, the Fuzzy ESVDF algorithm [118], [119], using the KDD-99 dataset [111], and four other datasets from UCI Irvine Machine Learning Repository [112]. The experimental results demonstrate that our approach can reduce training and testing times while retaining high classification accuracy.

For the second approach, the IDS classification scheme, we have implemented the different IDS types (AIDS, TIDS, and NIDS) by using the Fuzzy ESVDF approach to select the appropriate features set for each IDS type and validating their performance by using NN and SVM classifiers. Experimental results demonstrate that classifying IDS into different types improves system accuracy, efficiency (training time and testing time), and scalability even more. It is reduced from seven features (in case of using all attack types) to five features in AIDS, four features in TIDS, and four features in NIDS, as shown in Table 3.9.

Therefore, combining the two approaches, features selection and the IDS classification scheme, can build an efficient lightweight IDS that is small, powerful, and flexible enough to be used as a permanent element of a network security infrastructure.

# Chapter 4
# Collaborative Architecture for dIDS based on Lightweight IDSs

In this thesis, we propose a new architecture for IDS, called a Collaborative architecture for dIDS (C-dIDS) based on lightweight IDS modules, to overcome the heavy network traffic problem while improving system performance and scalability [153]. This architecture, C-dIDS, combines two main concepts. The first concept, the C-dIDS uses lightweight IDS (Chapter 3), where each detector (IDS module or host) uses small amounts of data in the detection process by applying two different approaches: features selection approach and IDS classification scheme.

For the first approach, a novel features selection approach called Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF) [118], [119] is used in order to improve system scalability in terms of reducing the number of needed features without affecting overall system performance.

The second approach uses a new IDS classification scheme by employing multiple specialized detectors in each layer of the network TCP/IP network model [152]. This helps in the collecting of efficient and useful information for dIDS, increasing the system's ability to detect different attack types and reducing the system's scalability. To integrate the system's IDS modules, we propose a new architecture, Collaborative dIDS, as a second concept used in this thesis. The C-IDS contains a single-level hierarchy dIDS with a non-central analyzer [154]. To make the detection decision for a specific IDS module in the system, this IDS needs to collaborate with the previous IDS in the lower level of the hierarchy only. This architecture improves overall system performance. It overcomes the distributed IDS (dIDS) limitations: the central management problem and the scalability bottleneck. Moreover, the cooperation between the IDS modules is done with less network load. In this chapter, first we explain the new architecture, C-dIDS. Then, we integrate the two concepts (lightweight and C-dIDS) in order to improve system performance, efficiency, scalability, generality, configurability, reliability, robustness, flexibility, and extensibility with minimum network load. To design this architecture, several experiments have been conducted which indicate that the proposed architecture can deliver satisfactory results.

This chapter splits into three sections. The first section (Section 4.1) describes the C-dIDS architecture. It starts by giving a brief background of dIDS. The proposed architecture C-IDS is then presented with some experimental results and discussion, followed by reviewing the summary of the

section. Section 4.2 describes the proposed system (C-dIDS based on lightweight IDS) with some experiments and results. Our conclusion is drawn in Section 4.3.

## 4.1 Collaborative Architecture for dIDS

Due to the many issues associated with monolithic architecture for an IDS, such as limited scalability, single point of failure, a lack of extensibility, much overload (computational bottleneck), vulnerable to subversion, difficult to configure or add capability among others, a distributed Intrusion Detection System (dIDS) is required to allocate multiple IDS modules across a network to monitor security events and to collect data. However, most dIDS architectures have two primary shortcomings. First, the central management and processing of data represents a single point of failure. Second, the scalability bottleneck often results in these systems suffering from a slow response time to new threats. In this section, we propose a new architecture to overcome these two limitations [154], called a Collaborative architecture for dIDS (C-dIDS). The C-dIDS contains a single-level hierarchy dIDS. To make the detection decision for a specific IDS module in the system, this IDS module needs to collaborate with the previous IDS in the lower level only. Coordinated deployment of multiple IDS promises to generate greater confidence in the results, and improve the coverage of intrusion detection. This can be accomplished with less network load (just one bit of information), which in turn improves system scalability. Moreover, by using single-level hierarchy, there is no central management and processing of data and so no chance of a single point of failure. We have examined the feasibility of our dIDS architecture by conducting several experiments using the DARPA dataset [111]. The experimental results indicate that the proposed architecture can deliver satisfactory system performance with less network load.

### 4.1.1 Distributed Intrusion Detection Systems

Many network-based and host-based IDS perform data collection and analysis centrally using a monolithic architecture (meaning that the data are collected by a single host and analyzed by a single module). This architecture suffers from significant problems that limit the performance of IDS [120], [121], [122]. First, a single point of failure: if an intruder can somehow prevent the IDS from working, the entire network is unprotected. Second, limited scalability: processing all the information on a single host implies a limit on the size of the network that can be monitored. After that limit, the central analyzer is unable to keep up with the flow of information. Third, a lack of extensibility: it is difficult to reconfigure or add capabilities to the IDS. Finally, the analysis of network data can be

flawed. As a result, intrusions can be conducted through several steps that occur on different hosts, and such intrusions consequently cannot be detected by a single IDS.

These problems make the area of IDS an attractive research field. In recent years, researchers have investigated different distributed approaches for IDS [42], [98], [99], [121]. The distributed IDS (dIDS) [77], [89] is one of several options that allow computation load and diagnostic responsibilities to be distributed throughout the network. It performs distributed data collection (and some pre-processing) by using modules distributed among different hosts, which monitor separately and communicate and cooperate with each other. The dIDS can provide the foundation for a complete solution to the complexities of real-time detection, while maintaining fault-resistant behaviour. In addition, each module can be added or removed from the system without altering other system components, as they operate independently. Moreover, the system's modules can be configured or upgraded without disturbing the rest of the system as long as their external interface remains the same. Nonetheless, the collected data are still shipped to a central location where they are analyzed by a monolithic engine. Also, it may result in a scalability bottleneck. To address these limitations, many techniques use a hierarchical structure [42], [43], [44], [46], [67] which describe a cooperative system without centralized analysis components. In these approaches, the local intrusion detection components look for local intrusions and pass the results of their analysis to the upper levels of the hierarchy. The components at the upper levels analyze the refined data from multiple lower level components and seek to establish a global view of the system state. This helps address scalability and allows a system to be deployed across large enterprise-scale networks. Moreover, it helps address the single point of failure problem, because if a higher node hierarchy should fail the lower tiers can typically continue to function, but with less detection capabilities.

The major disadvantages of hierarchical dIDS are the limited detection process (it limits the kind of intrusions that can be detected at the highest levels) and the network latency (the delay between each level in the architecture). Moreover, there is still one highest-level entity, which is the bottleneck of this system and leads to a single point of failure. The hierarchical structures usually give attackers the opportunity to harm the IDS by cutting off a control branch or even by taking out the root command. Our proposed approach overcomes these problems by eliminating the need for so much transferred data and speeding up the detection process [154]. Also, it addresses the single point of failure problem without losing any detection capability, and improves the overall scalability of the system.

This architecture is called Collaborative architecture for dIDS (C-dIDS). The C-dIDS contains a single-level hierarchy architecture. Each IDS module (host) in the system needs to receive one information bit from the IDS in the lower level and pass the results of its analysis to the IDS in the upper level. Therefore, to make the detection decision for each IDS module in the system, this IDS needs to get one bit of information (the analysis results) from the previous IDS in the lower level only, without proceeding to the more than one level or the root node. Thus, improves the overall system performance, avoids a single point of failure problem and speeds up the detection process with less network load.

### 4.1.2 Collaborative Architecture for dIDS

We propose a new architecture for distributed IDS (dIDS) called a "Collaborative architecture for dIDS" (C-dIDS) to overcome the single point of failure, heavy network traffic and network latency problems, while improving system performance. This architecture, C-dIDS, organizes the cooperation process between different IDS modules (hosts) that are distributed on different points in the network by using single-level hierarchy dIDS with non-central analyzer. Each IDS module (host) in the system needs to receive a single bit of information from the previous IDS module to make its own detection decision. This bit of information can be either zero (to indicate that the network traffic is normal), one (to indicate that the network traffic is attack), or two (to indicate that the network traffic is undefined, which means that the network traffic has two values: normal and attack). To make a detection decision for each IDS module, the IDS module needs to use this bit of information from the previous IDS module in the lower level with the six rules described by Table 4.1. Where T = zero (normal), F = one (attack), and U= two (Undefined); X1 is the detection value from the current IDS module and X2 is the detection value that it is sent from the previous IDS in the lower level of the hierarchy.

Table 4.1

The Composition Table for the Final Decision Results

| X1 | X2 | The Decision Result |
|---|---|---|
| T | T | T |
| T | F | U |
| T | U | T |
| F | T | U |
| F | F | F |
| F | U | F |

As it is shown in Table 1 that “X1” takes only two values either True (normal) or False (attack), because “X1” is the analysis result from the IDS module. However, “X2” can take three values: True (normal), False (attack) and Undefined; because “X2” is the information bit that comes from previous IDS module in the lower level. This bit is actually the outcome of Table 4.1, and, as it shown, it has three values.

Then, only normal and undefined traffic are allowed to pass to the next IDS module in the upper level; attack traffic is blocked and denied passage to the next stage. Moreover, for initializing the information bit at the first IDS module (host), there are three different scenarios. The first scenario is to initialize the bit (X2) with one (attack: F). Figure 4.1 describes this scenario. The second scenario is to initialize the bit with two (undefined: U). Figure 4.2 describes this scenario. The third scenario is to initialize the bit with zero (normal: T). Figure 4.3 describes this scenario. (Suppose the number of IDS modules equals three).

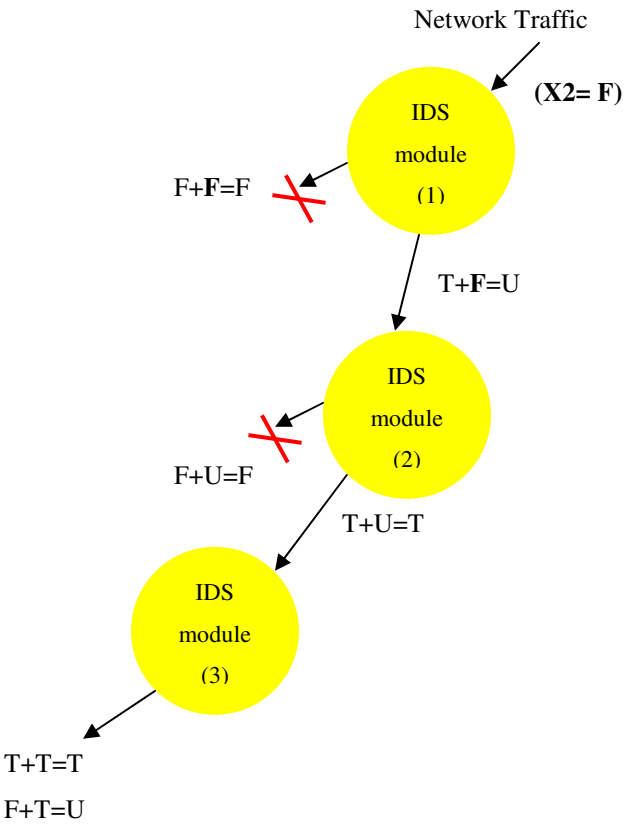



Figure 4.1 The first scenario for C-dIDS architecture

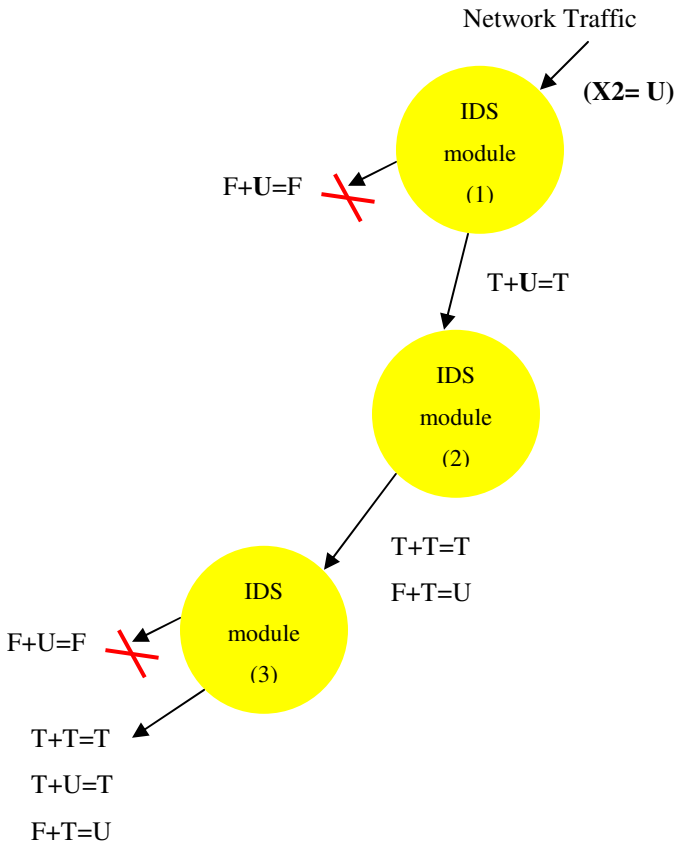



Figure 4.2 The second scenario for C-dIDS architecture

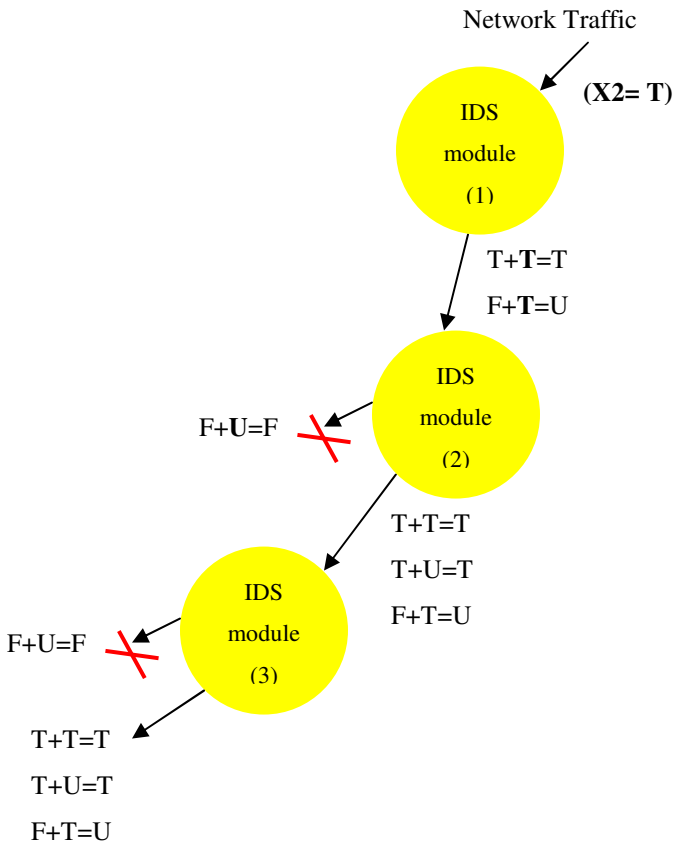



Figure 4.3 The third scenario for C-dIDS architecture

As shown in the above figures, the initialization of the information bit (X2) affects the system behaviour in the first stages only. In scenario (1), after the system has filtered the traffic it will allow for only the normal to pass on to the first two stages; then it will allow all traffic to pass in the third stage, and after that it will allow the normal and undefined traffic to pass the rest of the stages. For scenario (2), the system at the first stage will allow the normal to pass only. For the second stage, it will allow all traffic to pass to the next stage, and then it will allow the normal and undefined traffic to pass the rest of the stages. Finally, in scenario (3), for the first stage it will allow all traffic to pass and then only normal and undefined traffic will be allowed for the rest of the stages.

Another feature of this C-dIDS architecture is its flexibility in terms of its ability to be converted to other distributed architectures (non-cooperative dIDS and central analyzer dIDS) by changing the value of the information bit (the bit that it is received from the previous IDS in the lower level) to fix value either zero (normal/T) or one (attack/F). If the information bit (X2) has a value of one, the

system's structure becomes like a non-cooperative dIDS; and if it has a value of zero, the system's structure becomes like a central analyzer dIDS.

The non-cooperative dIDS architecture is a distributed architecture where each IDS module does it own detection decision without any cooperation with others. The C-dIDS can be converted to this architecture by fixing the information bit (X2) value to one (attack/F) always as it is shown in Figure4.4.

In the central analyzer dIDS, each IDS module does its detection decision. At the end, all IDS modules will send their results to the central analyzer, which will aggregate all decisions into a single decision using an aggregation technique. The C-dIDS can be converted to this architecture by fixing the information bit (X2) with zero value (normal/T), as it is shown in Figure 4.5.

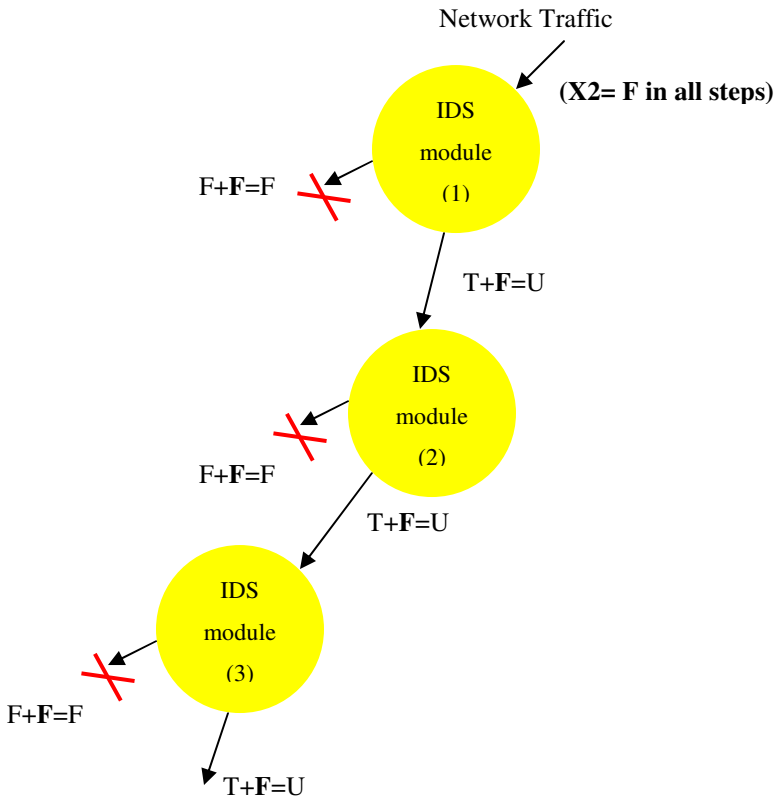



Figure 4.4 The non-cooperative architecture for C-dIDS

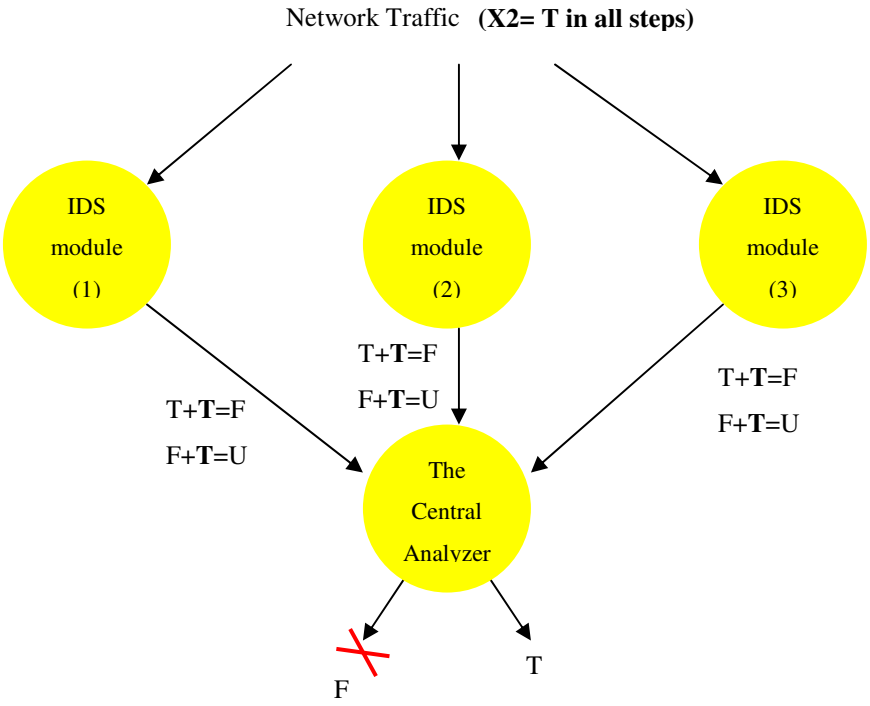



Figure 4.4 The non-cooperative architecture for C-dIDS

As a result, the proposed architecture, C-dIDS, reduces the network traffic load while improving overall system performance. Moreover, there is no central management and processing of data so there is no chance for a single point of failure. In addition, C-dIDS is a flexible system. It can be converted to other dIDS architectures: non-cooperative dIDS or central analyzer dIDS. The experimental results in the next section will prove some of these features.

### 4.1.3 Experiments and Results

To evaluate the performance of our proposed approach, we compare it with two other architectures- central analyzer dIDS and non-cooperative dIDS - by using the DARPA KDD-99 benchmark dataset [111]. In this section, we initially describe the contents of the DARPA KDD-99 dataset and the experimental settings, followed by some experimental results and discussion.

#### *Datasets Description*

We use the same dataset (KDD-99 dataset [111]) as employed in the previous section (Section 3.3.1). In these experiments, we pick two different datasets for training and testing purposes. Each dataset

contains 6000 samples, of which 3000 are normal samples (50 %) and 3000 are attack samples (50 %) (i.e., the total number of samples equals 12000).

### *Experimental Settings*

Our experiment has two main steps. In the first step, we build three IDS modules (hosts). Each module is implemented using different technique. Secondly, we integrate these different IDS modules using three different architectures: C-dIDS, non-cooperative dIDS, and central analyzer dIDS.

For the first step and for building three IDS modules, we use: Back Propagation (PB) Neural Networks to implement IDS module (1), Radial Basis Function (RBF) neural network to implement IDS module (2), and for IDS module (3) we use Support Vector Machines (SVMs). (Note: there is no reason behind picking these SC tools to build the IDS modules; our focus here to build three IDS modules regardless of the tools that are used)

For implementing BP NN, we use the function "newff" from the MATLAB toolbox with three layers (an input layer with 41 neurons, a hidden layer with six neurons, and an output layer with one neuron), sigmoidal activation function, performance function "MSE", 30 epochs and 0.001 learning rate. For RBF NN, we use the function "newrb" from MATLAB toolbox with goal value equal to zero, spread value equal to one, and maximum number of neurons equal to fifty. For SVM implementation, we use the simpleSVM library for SVM [113]. The crossover parameters selection of the SVM includes a range of basic SVM parameters, various kernel functions, and their performance arguments. In our experiments, we set the crossover parameters as follows: $C$ can take one of these values 1, 100, 5000, or 10000. The kernel functions for SVM were taken as linear and radial basis kernels. The polynomial kernel with degree 1 and 2 and the coefficient (scale) can be 0.5, 2, 3, or 4. $\sigma$ in a radial basis kernel at either 0.5, 1, 2, or 3.

For the second step, we integrate the three modules (IDS module (1), IDS module (2), and IDS module (3)) using three different dIDS architectures: C-dIDS, non-cooperative dIDS, and central analyzer dIDS.

In the C-dIDS architecture, each IDS module makes its detection decision by using one bit of information from the previous IDS in the lower level of hierarchy (as it is explained in Section 4.1.2). Network traffic first enters the IDS module (1) that will divide the traffic into normal and attack, which are the values of "X1". Then IDS module (1) applies the six rules that are mentioned in Table4.1 on its result analysis "X1" and the information bit "X2" sent from a previous IDS module to

make the final decision. After that, it only allows normal and undefined traffic to pass to the next module. Normal and undefined traffic will again be scrutinized at the IDS module (2) which will divide the traffic once more into normal and attack, and also applies the six rules on them to get the final analysis results. Then IDS module (2) allows only normal and undefined traffic to pass on to the next module. The same situation applies to the IDS module (3).

The non-cooperative dIDS is a distributed architecture where each IDS module does its own detection decision, without any cooperation with others. In our case, we have three different IDS modules. The network traffic first enters the IDS module (1) which will divide the traffic into normal and attack, and allow only the normal traffic to pass to the next IDS module in the upper level; the remaining attack traffic will be blocked. Normal traffic will again be scrutinized at the next IDS module, which will divide the traffic once more into normal and attack, and allow only the normal traffic to pass on to the next IDS module. This process will happen repeatedly until the filtered network traffic is received at the last IDS module.

In the central analyzer architecture, each IDS module in the system makes its own detection decision. At the end, all IDS modules will send their results to the central analyzer, which will aggregate all decisions into one final decision using an aggregation technique. In our case, we have three IDS modules (three votes). To aggregate these votes, we use a vote aggregation method. So, there might be three normals, or two normals and one attack; in both cases, “normal” wins. Or there can be three attacks, or two attacks and one normal; in both cases, “attack” wins.

Each experiment is repeated ten times by randomly selecting 40 % of samples as the test data; the remaining 60 % are used as the training data.

### *Experimental Results*

The comparisons of the different dIDS architectures {C-dIDS [C-dIDS (1), C-dIDS (2) and C-dIDS (3)], non-cooperative dIDS, and central analyzer dIDS} are presented in Table 4.1 and Table 4.2 respectively. These comparisons are done with respect to different performance indicators: FPR, CR, training time, and testing time (Table 4.2), the average amount of traffic that enters each IDS module, and the total amount of traffic (Table 4.3). (Note: C-dIDS (1) refers to the first scenario for C-dIDS, C-dIDS (2) refers to the second scenario for C-dIDS, and C-dIDS (3) refers to the third scenario for C-dIDS).

Table 4.2

The Comparison between the three architectures: C-dIDS, non-cooperative, and central analyzer in terms of FPR, CR, and efficiency

| **Architecture** | **FPR (%)** | **CR (%)** | **Training Time (sec)** | **Testing Time (sec)** |
|---|---|---|---|---|
| C-dIDS (1) | $1.00 \times 10^{-6}$ | 99.304 | 402.822 | 0.233 |
| C-dIDS (2) | $1.00 \times 10^{-6}$ | 99.609 | 402.822 | 0.247 |
| C-dIDS (3) | $0.84 \times 10^{-6}$ | 99.711 | 402.823 | 0.390 |
| non-cooperative | $1.2 \times 10^{-6}$ | 99.292 | 402.822 | 0.239 |
| central analyzer | 0.336574 | 99.774 | 402.822 | 0.444 |

Table 4.3

The Comparison between the three architectures: C-dIDS, non-cooperative, and central analyzer in terms of amount of traffic

| **Architecture** | **Traffic (IDS1)** | **Traffic (IDS2)** | **Traffic (IDS3)** | **Total traffic** |
|---|---|---|---|---|
| C-dIDS (1) | (2391.2 x 41) 98039.2 + | (1345.2 x 41) 55153.2 + | (1200.6 x 41) 49224.6 + | 207354 |
| C-dIDS (2) | (2391.2 x 41) 98039.2 + | (1345.2 x 41) 55153.2 + | (1280.7 x 41) 52508.7 + | 210718.2 |
| C-dIDS (3) | (2391.2 x 41) 98039.2 + | (2391.2 x 41) 98039.2 + | (1199.6 x 41) 49183.6 + | 251244 |
| non-cooperative | (2391.2 x 41) 98039.2 | (1345.2 x 41) 55153.2 | (1200.6 x 41) 49224.6 | 202417 |
| central analyzer | (2391.2 x 41) 98039.2 | (2391.2 x 41) 98039.2 | (2391.2 x 41) 98039.2 | 294117.6 |

Note: "+" means the amount of traffic plus the information bits (the bits that are added from the cooperation process between the modules). For example "2391.2 x 41 = 98039.2 +", means that 98039.2 + 2391.2 = 100430.4. The total traffic amount includes this additional amount (the added information bits), and 41 represents the number of the information bits in each record of the network traffic.

### *Discussion*

As shown in Table 4.2, comparison between C-dIDS (C-dIDS (1), C-dIDS (2), C-dIDS (3)) and non-cooperative dIDS reveals a significant improvement in terms of CR with similar FPR is maintained. The C-dIDS (1), C-dIDS (2), and non-cooperative dIDS have nearly the same testing time. However, the testing time is increased in C-dIDS (3) because the C-dIDS (3) allows more traffic to pass at the first stage of detection, as is shown in Table 4.3. In general the proposed architecture improves the system efficiency (time) without creating a heavy network load on the system or reducing system accuracy. Moreover, the non-cooperative dIDS represents dIDS with no central analyzer and without any cooperation between the IDS modules. Each IDS module works independently and filters network traffic according to its own decision, which causes a reduction in overall system accuracy, because each layer adds some portion of error through blocking attack records, which may contain some normal records (False Positive); also, aggregating more than one decision is better than an individual decision. The C-dIDS mitigates these limitations by allowing more traffic to pass to the next IDS module in the system (it allows the normal and undefined traffic instead of only the normal traffic) and cooperating with the other IDS module to make the final detection decision.

Comparing C-dIDS (C-dIDS (1), C-dIDS (2), and C-dIDS (3)) with the central analyzer dIDS, the central analyzer dIDS outperforms the C-dIDS in terms of CR; however, it has the largest FPR value and testing time. In the central analyzer dIDS, each IDS module works independently, and the result of each module is aggregated in order to generate more global alerts. Therefore, all data will be shipped to all dIDS modules, and then sent to a central location for aggregating the alerts, which causes heavy network traffic as shown in Table 4.3. The total amount of traffic is 294117.6, which is higher than other architectures. Moreover, the central analyzer dIDS suffers from the single point of failure problem, and that may prevent IDS from working and cause the entire network to experience a loss of protection.

Comparing C-dIDS (1), C-dIDS (2), and C-dIDS (3) shows that allowing more traffic to pass between system's IDS modules at the first stages can improve the overall system performance (as shown in Table 4.2) while increasing the testing time (as shown in Table 4.2) and the system load (as shown in Table 4.3). Moreover, it is obvious from the above figures that dIDS (1) works as a non-cooperative architecture in the first two stages. dIDS (2) also works as a non-cooperative architecture, but only in the first stage. For dIDS (3), the system does not do that at all, which means that changing the value of initialization bit can only affect the system in the first stages. For the rest of the stages, all scenarios will follow the same behaviour. As a result, there is no preference between these three

scenarios. Each one has its own benefits, depending on user requirements. The C-dIDS (1) scenario can be used in applications in which time plays a critical role (it has the least testing time = 0.233). On the other hand, if system accuracy is the most important issue in the application, the C-dIDS (3) scenario is recommended (it has the best CR = 99.711 and FPR = 0.84 x $10^{-6}$). Accordingly, if time and system accuracy have the same priority, dIDS (2) is recommended. As shown in Table 4.2 and Table 4.3, the C-dIDS (1) has the best value in terms of the traffic amount and the testing time, and C-dIDS (3) has the best system accuracy among other C-dIDS scenarios. The application type will then determine which scenario is the most appropriate. In general, the idea of C-dIDS is to improve the non-cooperative architecture by allowing more traffic to pass between the system's IDS modules. Instead of allowing only the normal traffic to pass, the C-dIDS allows normal and undefined traffic. Moreover, cooperating processes between system modules can improve the overall system accuracy.

In summary, these results demonstrate the feasibility of the proposed architecture (C-dIDS). The C-dIDS seems to be the most appropriate approach because it uses a non-central analyzer dIDS, and it allows the modules to cooperate with less network load (one bit of information through single-level hierarchy dIDS). It is also shown that the C-dIDS is a flexible system; it can be converted to either a central analyzer dIDS or a non-cooperative dIDS by changing only the value of the information bit.

### 4.1.4 Summary

The most common shortcomings in existing distributed Intrusion Detection System (dIDS) architectures are that they are built around a central management that does the aggregation and processing of the system's alerts. A heavy network load results in very large amounts of data being transmitted between the detectors (hosts). This section presents a novel architecture for dIDS to overcome these limitations, called Collaborative architecture for dIDS (C-dIDS) [154]. The C-dIDS demonstrates that detection IDS modules can be run in a distributed fashion, with each one running independently of the others while cooperating and communicating to provide a truly distributed detection mechanism with no single point of failure. There is no central processing location; all IDS modules process the information available to them independently and relay any suspicious activity to other IDS modules on the network. The C-dIDS is based on the idea of independent IDS cooperating to detect different attack types across the network. Each IDS module makes its own traffic analysis while cooperating with other detectors to make the final detection decision. The main goals of this architecture are to reduce network traffic load while achieving better intrusion detection.

By using single-level hierarchy dIDS with a non-central analyzer, each IDS module in the system needs to communicate with other IDS modules by transferring one bit of information. Cooperating with other IDS modules (detectors or hosts) can improve the system's ability to detect attacks that might not be detectable if each IDS module was examined individually, with low network load. Moreover, by using single hierarchy level, there is no central management and processing of data so there is no chance for a single point of failure. In addition, fewer data are transferred between these modules (just one bit of information). The C-dIDS is a flexible system. It can be used as a central analyzer dIDS or a non-cooperative dIDS simply by fixing the value of its information bit to either one or zero.

We evaluate the proposed architecture, C-dIDS, by comparing it with other dIDS architectures: non-cooperative dIDS and central analyzer dIDS. The experimental results illustrate that the C-dIDS is a suitable architecture in terms of system performance and network load.

## 4.2 Collaborative Architecture for dIDS based on Lightweight IDS

To even further improve the efficiency of the C-dIDS in terms of system scalability and network load, we use lightweight IDS as system detectors (hosts). So, the proposed architecture of the dIDS will integrate two different concepts: lightweight IDS and a distributed collaborative architecture.

In the first concept, lightweight IDS (Chapter 3), the detection process uses fewer data for the detection process by using lightweight IDS modules. The lightweight IDS is a small, flexible, and highly capable system that is in use around the world on both large and small networks. It accomplishes its essential tasks with minimal data, and it is dynamically updatable and upgradable, simpler, and faster to transport (due to its smaller size). To build a lightweight IDS module, we need to reduce the amount of data/features that are needed to accomplish the detection process. Most researchers in this area use one of the features selection approaches to design a lightweight IDS, which is considered to be inefficient in most cases. In this thesis, we build a lightweight IDS by integrating two different approaches: features selection and an IDS classification scheme. For the first approach, we apply a novel features selection algorithm called Fuzzy Enhancing Support Vector Decision Function (Fuzzy ESVDF). The Fuzzy ESVDF is an iterative algorithm based on Support Vector Decision Function (SVDF) and Forward Selection (FS) with a fuzzy inferencing model [118], [119]. Using the Soft Computing (SC) approach for IDS features selection is suitable for handling such subjective estimates, due to their high performance, low solution cost, fast recognition and classification of different attacks, and ability to generalize from learned data. In the second approach,

we present a new IDS classification scheme based on a TCP/IP network model. In this scheme, specialized IDS are placed at each of the four layers of a TCP/IP network model (Application layer IDS, Transport layer IDS, Network layer IDS, and Link layer IDS) to detect specific types of attack corresponding to each layer. This scheme would enhance an organization's ability to detect most types of attack by identifying correct locations to place an IDS for the following reasons. First, each TCP/IP layer has different vulnerabilities, security challenges and types of attacks. The studies in [115], [116], [19], [117] showed that the choice of network features for IDS depends on the network attack type to be detected. Network attacks can be categorized into four major types: (1) Application Layer attacks, (2) Transport Layer attacks, (3) Network Layer attacks, and (4) Link Layer attacks. The IDS can also be categorized into AIDS, TIDS, NIDS, and LIDS. Second, as is known, firewalls operate at different TCP/IP network layers by using different criteria to restrict traffic, but this is a long step from running an entirely secure network. Because of that, IDS must be allocated as a second line of defense behind the firewalls. Third, the attacks usually gain access to the network through the network devices distributed through different TCP/IP network layers as entry points, and in order to be able to adequately address security, all possible avenues of entry must be evaluated and secured. So, IDS must be allocated at these entry points or network devices. Finally, splitting the detection process into different levels and stages reduces the computation load on the system and improves its scalability and performance. Accordingly, the proposed lightweight IDS improves system accuracy, efficiency, scalability, generality, extendibility, flexibility, and configurability.

Due to the increasing connectivity of heterogeneous computer systems and the rapid growth, sophistication, coordination and cooperation of attack tools and strategies, using distributed IDS (dIDS) becomes essential in designing an IDS. The dIDS consists of multiple entities working independently and allows changes to these entities without any modifications made to other entities. The dIDS is capable of improving system performance, scalability, and extensibility, and can provide the foundation for a complete solution to the complexities of real-time detection, while maintaining fault-resistant behaviour. However, it suffers from a number of limitations, such as a scalability bottleneck, a limited detection process, and network latency.

In this thesis, we propose a novel collaborative architecture for distributed intrusion detection as a second concept (Section 4.1) in order to overcome some of the limitations of the current approaches in dIDS. This Collaborative distributed IDS (C-dIDS) is based on a single-level hierarchy dIDS with a non-central analyzer. Each IDS module in the system needs to receive a single bit of information from the previous IDS module to make its own detection decision, without proceeding to the root

node or more than one IDS. Transferred data can be dispatched between the detectors with only crucial data (just one bit of information), which will reduce network load. Moreover, data collection and information analysis are performed locally without referring to the central management unit. Therefore, there is no scalability bottleneck or single point of failure. In addition, it is capable of improving accuracy, real-time performance, efficiency, flexibility, adaptability, extensibility, robustness, and fault tolerance, as explained in Section 4.1.

In this section, we integrate the above two concepts (lightweight IDS and Collaborative dIDS) in order to improve the overall system performance. Using a lightweight IDS can improve system efficiency, accuracy, scalability, generality, extendibility, flexibility, and configurability. The second concept, C-dIDS, can also improve system scalability, extendibility, configurability, and flexibility. In addition, it can improve system reliability and robustness with minimum network load.

### 4.2.1 Experiments and Results

To evaluate the performance of our proposed approach, we compare it with other architectures by using the DARPA KDD-99 benchmark dataset [111]. In this sub-section, we initially describe the contents of the DARPA KDD-99 dataset and the experimental settings, followed by some experimental results and discussions.

#### *Datasets Description*

We use the same dataset (KDD-99 dataset [111]) as already used in the previous section (Section 3.3.1). In these experiments, we pick two different datasets for training and testing purposes. Each dataset contains 6000 samples; of which 3000 are normal samples (50 %) and 3000 are attack samples (50 %) (i.e., the total number of samples equals 12000).

#### *Experimental Settings*

Our experiment has two main steps. In the first step, we build three IDS modules. Secondly, we integrate these different IDS modules to build different dIDS architectures. We build four architectures: C-dIDS with specialized IDS module, C-dIDS with non-specialized IDS module, non-cooperative dIDS with non-specialized IDS module, and central analyzer dIDS with non-specialized IDS module. (Note: specialized IDS modules means that we use Network layer IDS (NIDS), Transport layer IDS (TIDS), and Application layer IDS (AIDS) that we had already built in Section 3.2, and a non-specialized IDS module means that IDS modules use all 41 features).

For the first step, we use the results of previous experiments (Section 3.2) to build the C-dIDS with specialized IDS modules. The specialized IDS modules will be NIDS, TIDS, and AIDS. For other architectures, we use non-specialized IDS modules.

For the second step, we integrate the three modules (NIDS, TIDS, and AIDS) to build a C-dIDS with a specialized IDS module. Also, we integrate the non-specialized IDS modules to build the other architectures: C-dIDS with non-specialized IDS module, non-cooperative dIDS with non-specialized IDS module, and central analyzer dIDS with non-specialized IDS module. The different architectures are explained in Section 4.1.2.

Each experiment is repeated ten times by randomly selecting 40 % of the samples as the test data; the remaining 60 % are used as the training data.

### *Experimental Results*

For C-dIDS with specialized modules, we apply the Fuzzy ESVDF on 41 features [118], [119] to select the best features set for each type of IDS module: AIDS, TIDS, and NIDS (Section 3.2). The resulting features sets for AIDS, TIDS, and NIDS are presented in Table 3.9. The comparisons of the different architectures - C-dIDS with specialized (we will used scenario (2) of the C-dIDS), C-dIDS with non-specialized (we will used scenario (2) of the C-dIDS), non-cooperative with non-specialized dIDS, and central analyzer with non-specialized dIDS - are presented in Table 4.4 and Table 4.5. The comparison of the different dIDS architectures is done with respect to different performance indicators: FPR, DR, CR, training time, and testing time (Table 4.3), the average amount of traffic that enters each IDS module, and the total amount traffic (Table 4.4).

For the sake of simplicity, we re-named each architecture as follows:

- SC-dIDS: C-dIDS (2) with specialized IDS
- NC-dIDS: C-dIDS (2) with non-specialized IDS
- non-cooperative : non-cooperative with non- specialized IDS
- central analyzer: central analyzer with non- specialized IDS

Table 4.4

Comparison between the four structures in terms of FPR, DR, CR, and efficiency

| **Architecture** | **FPR (%)** | **DR (%)** | **CR (%)** | **Training Time (sec)** | **Testing Time (sec)** |
|---|---|---|---|---|---|
| SC-dIDS | $9.2 \times 10^{-5}$ | 96.642 | 97.312 | 572.372 | 0.103 |
| NC-dIDS | $8.3 \times 10^{-6}$ | 96.528 | 97.762 | 3099.00 | 0.143 |
| non-cooperative | $8.4 \times 10^{-6}$ | 96.535 | 97.486 | 3099.00 | 0.128 |
| central analyzer | 0.4384 | 99.768 | 99.703 | 3099.00 | 0.178 |

Table 4.5

Comparison between the four architectures in terms of passed traffic

| **Architecture** | **Traffic (IDS1)** | **Traffic (IDS2)** | **Traffic (IDS3)** | **Total traffic** |
|---|---|---|---|---|
| SC-dIDS | (2385.3x4) 9541.2 + | (1330.5x 4) 5322 + | (1200.5x5) 6002.5 + | 25782 |
| NC-dIDS | (2385.3x41) 97797.3 + | (1328.3x41) 54460.3 + | (1220.2x41) 50028.2 + | 207219.6 |
| non-cooperative | (2385.3x41) 97797.3 | (1330.5x41) 54550.5 | (1182.5x41) 48482.5 | 200830.3 |
| central analyzer | (2385.3x41) 97797.3 | (2385.3x41) 97797.3 | (2385.3x41) 97797.3 | 293391.9 |

Note: "+" means that the amount of the traffic plus the information bits (the bits that are added from the cooperation process between the modules). For example "2385.3 x 4 = 9541.2 +", means that 9541.2 + 2385.3 = 11926.5. The total traffic amount reflects this additional amount (the added information bits).

### *Discussion*

As shown in Table 4.4, comparison between SC-dIDS and NC-dIDS reveals a significant improvement in terms of training time (it is reduced from 3099.0 sec to 572.372 sec) and testing time (it is reduced from 0.143 sec to 0.103 sec) while keeping nearly the same CR, DR, and FPR in both cases. Also, Table 4.5 shows significant improvement in terms of the amount of the traffic between the modules in SC-dIDS (25782 information bits) and NC-dIDS (207219.6 information bits). In

general, the results indicate that using a lightweight IDS module can improve overall system efficiency (training time and testing time) and system load without affecting system accuracy.

In comparing SC-dIDS with non-cooperative architecture, we find that although SC-dIDS does not use all 41 features, both non-cooperative dIDS and SC-dIDS architecture have nearly the same DR, CR, and FPR. On the other hand, Table 4.4 shows the obvious improvement in training time and testing time. Also, Table 4.5 reveals significant improvement in terms of the total amount of traffic. The SC-dIDS has 25782 information bits, while non-cooperative architecture has 200830.3 information bits. As a result, the proposed architecture (SC-dIDS) has the ability to improve system efficiency with a high accuracy value and a lighter network load.

Comparing SC-dIDS with a central analyzer architecture, the central analyzer dIDS outperforms the SC-dIDS in terms of CR and DR; however, it has the largest FPR. Also, its training and testing time are greater than the training and testing time in the case of SC-dIDS, as shown in Table 4.3. Table 4.4 shows that the traffic passed between the IDS module in the central analyzer architecture is higher than SC-dIDS; this is because all data in the central analyzer will be shipped to all dIDS modules and then sent to a central location for aggregating the alerts. Accordingly, a central analyzer architecture causes heavy network traffic on the system. This is in addition to the central management problem.

In summary, the proposed architecture, C-dIDS with a specialized IDS module (SC-dIDS), can improve the overall system efficiency (training time and testing time), scalability, and network load while still delivering satisfactory system accuracy.

## 4.3 Conclusion

The most common shortcomings in the current IDS are low accuracy, low efficiency, and limited scalability. This chapter presents a novel IDS distributed architecture –Collaborative Distributed Intrusion Detection System (C-dIDS) based on lightweight IDS modules [153]. The C-dIDS is based on the idea of lightweight, independent IDS cooperating to detect different attack types in each TCP/IP network layer, by employing multiple specialized detectors at various layers of the network TCP/IP model with a cooperative architecture. Each detector is specialized to detect different types of attacks by cooperating with other detectors to increase user confidence in the alert. The main goals of this architecture are to reduce network traffic load while improving system performance (accuracy and efficiency). These goals are accomplished through use of two different concepts.

First, using lightweight IDS modules (Chapter 3), the detection process uses less data by applying two different approaches: features selection and an IDS classification scheme. For the features selection approach (Section 3.1), we use a novel algorithm called the Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF) [118], [119]. The Fuzzy ESVDF integrates the Support Vector Decision Function (SVDF) and Forward Selection (FS) approaches with the fuzzy inferencing model to select the most appropriate features set for the IDS. It produces an efficient features set by using a fast and simple approach. The second approach is the IDS classification scheme (Section 3.2). This scheme employs multiple specialized detectors in each layer of the network TCP/IP model, and data can be collected from multiple sources [152]. Thus, combining the best characteristics of traditional host-based, network-based, and router-based IDS can improve the overall performance and scalability of the system. Moreover, this categorization gives the architecture extended and maintained ability.

Secondly, using a single-level hierarchy dIDS with a non-central analyzer (Section 4.1), each IDS module in the system needs to communicate with another IDS module by transferring one bit of information [154]. Cooperating with other IDS modules (detectors) can improve the system's ability to detect attacks that might not be detectable if each of the IDS was examined individually, with less network load. Moreover, by using single hierarchy level, there is no central management and processing of data, so there is no chance for a single point of failure; in addition, fewer data are transferred between these modules (just one bit of information). The C-dIDS is a flexible system. It can be used as a central analyzer dIDS or a non-cooperative dIDS simply by fixing the value of its information bit to either one or zero, as explained in Section 4.1.2. By comparing the proposed architecture with other architectures, we illustrate that the C-dIDS is a suitable architecture in terms of system accuracy, efficiency, scalability, and system load.

By integrating the above two concepts, lightweight IDS and C-dIDS, the overall performance of the IDS is improved. The experimental results show that a C-dIDS based on lightweight IDS modules improves overall system performance, efficiency (training time and testing time), and scalability without creating a heavy system load.

# Chapter 5
# Conclusions and Future Work

In this chapter, we briefly present a summary and contribution of the thesis in Section 5.1, followed by future work in Section 5.2.

## 5.1 Summary of Results and Thesis Contribution

In this thesis, we have primarily investigated the intrusion detection problem. Current IDS usually have several major shortcomings such as low accuracy, low real-time performance (low efficiency), and limited scalability. In particular, we have proposed a novel IDS architecture –Collaborative Distributed Intrusion Detection System (C-dIDS) based on lightweight IDS modules— that integrates two different concepts in order to work around these limitations [153]. First, the C-dIDS uses lightweight IDS, where each detector (IDS module) uses smaller amounts of data in the detection process by using two approaches: a features selection approach, and an IDS classification scheme. For the former, we apply a Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF) as a feature selection technique, which ensures that this technique will improve system efficiency, scalability, and reduce the network traffic load while retaining high classification accuracy.

The second approach is the IDS classification scheme. This scheme employs multiple specialized detectors in each layer of the network TCP/IP model, which helps in the collection of efficient and useful information for the dIDS, increasing the system's ability to detect different attack types and reducing the system's scalability. The second concept is accomplished by using Collaborative architecture (C-dIDS) for the IDS. The C-dIDS contains a single-level hierarchy dIDS with a non-central analyzer. To make the detection decision for a specific IDS module in the system, this IDS needs to collaborate with the previous IDS only, improving system accuracy without increasing the traffic load. Moreover, this architecture protects the system from single point of failure and the scalability bottleneck.

For the first approach, using lightweight IDS modules (detectors) (Chapter 3), the detection process uses fewer amounts of data for the detection process by employing two different concepts. The first approach is applying a novel feature selection technique (Fuzzy Enhanced Support Vector Decision Function) [118], [119]. The Fuzzy SVDF is based on a Support Vector Decision Function (SVDF) and Forward Selection (FS) with a fuzzy inferencing model. It is an iterative algorithm, where each iteration consists of two steps: feature ranking and feature selecting. Taking feature ranking first, the

Support Vector Decision Function (SVDF) is evaluated to rank each specified candidate feature. Next, feature selecting, a Forward Selection approach (FS) is applied with the fuzzy inferencing model to select the features according to a set of rules based on a comparison of performance.

To examine the feasibility of our approach, we conduct several experiments and comparisons. For evaluating the performance of the Fuzzy SVDF approach, we compare it with [34], [105] approaches by using the DARPA KDD-99 benchmark dataset [111]. In addition, we select four smaller datasets from the UCI databases [112] to evaluate the proposed approach (Fuzzy SVDF) in different domains, and its behaviour with a different number of features (each dataset has a varying number of features). Also, we use two different classifiers (SVM and NN) to evaluate the resulted features set. The experimental results demonstrate the feasibility of the proposed approach. The proposed approach gives the best performance in terms of training and testing times while retaining high classification accuracy, allowing this approach to be used in a real-time environment. In addition, this approach is considered to be a features selection approach regardless of the type of classifier used, making this approach a suitable features selection method for other applications rather than an IDS.

For the second approach, the IDS classification scheme, by employing multiple specialized detectors in each layer of the network TCP/IP model, data can be collected from multiple sources [152]. Thus, combining the best characteristics of traditional host-based, network-based, and router-based IDS can improve the overall performance and scalability of the system. Moreover, this architecture can be easily extended and maintained. We design three different types of IDS: NIDS, TIDS, and AIDS (LIDS is not included) by using Fuzzy ESVDF. Several experiments have been conducted to evaluate the effects of categorizing the IDS into these different types. We compare the performance of the specialized IDS modules (NIDS, TIDS, and AIDS) with the performance of the IDS that is designed to detect any attacks by using two different classifiers, NN and SVM. Moreover, we swap the features between the three different specialized IDS modules to evaluate the affect of each feature on the detection process. The experimental results indicate that each IDS type is subject to its own "custom" attacks and therefore needs its own custom protection. Also, categorizing IDS into different types can improve the overall system performance (accuracy and efficacy) and scalability.

For the second concept (Section 4.1), a distributed collaborative architecture, we propose a new architecture called a Collaborative architecture for dIDS (C-dIDS) [154]. The C-dIDS contains a single-level hierarchy dIDS with a non-central analyzer. Each IDS module in the system needs to communicate with another IDS module by transferring one bit of information. Cooperating with other

IDS modules (detectors) can improve the system's ability to detect attacks that might not be detectable if each module of the IDS was examined individually, with less network load. Moreover, by using single hierarchy level, there is no central management and processing of data, so there is no chance for a single point of failure. We have examined the feasibility of our dIDS architecture by conducting several experiments using the DARPA dataset, and compared the proposed architecture (C-dIDS) with other dIDS architectures: non-cooperative dIDS and central analyzer dIDS architecture. For each of these architectures, we use three different IDS modules. Module (1) is implemented by using Back Propagation (PB) Neural Networks; Module (2) employs a Radial Basis Function (RBF) Neural Network; Module (3) is implemented by Support Vector Machines (SVM). The experimental results show that the C-dIDS is a suitable architecture in terms of system performance and network load, as it allows the modules to cooperate with less network load (one bit of information through a single-level hierarchy dIDS). In addition, the C-dIDS is a flexible system. It can be used as a central analyzer dIDS or a non-cooperative dIDS simply by fixing the value of its information bit to either one or zero.

We join the above two concepts (Section 4.2): lightweight IDS (Chapter 3), and distributed collaborative architecture (Section 4.1) in order to improve system accuracy, efficiency, and scalability while reducing the overall system load [153]. Therefore, many IDS modules can be installed and work in a collaborative manner without creating a heavy network load. Each one is concerned with some specific part of the network. It is then necessary to make them cooperate to achieve a global vision of the intrusion, while avoiding a single point of failure. Another point that vindicates this cooperative approach is the possibility to combine IDS that work differently. We implement the C-dIDS architecture using three specialized IDS modules: NIDS, TIDS, and AIDS that were already implemented in Section 3.2. For evaluating the proposed architectures, we compare the proposed approach (C-dIDS) with non-specialized C-dIDS, non-cooperative, and central analyzer architecture. The experimental results demonstrate that the proposed architecture can improve the overall system efficiency (training time and testing time) and scalability while it delivers satisfactory system accuracy.

In summary, these results demonstrate the feasibility of the proposed architecture (C-dIDS based on lightweight IDS modules). By integrating the two concepts (lightweight IDS and distributed collaborative architecture for the IDS), the system accuracy, efficiency (training and testing time), and scalability are improved without creating extra network load on the system. It provides a layer of

defense which monitors network traffic for predefined suspicious activity or patterns, and alerts system administrators when potential hostile traffic is detected. Other advantages accrue as well.

First, it can be easily extended and maintained. Each module can be added or removed from the system without altering other system components, because intrusion detection processes are independent, so existing processes do not need to be modified when a new intrusion detection process is added.

Second, splitting the detection onto different levels (or stages) in the network reduces the intrusion's influences, which reduces any damage that may occur.

Third, the proposed architecture improves the reliability of the system. Because it distributes the detection onto different levels, an intruder successful in attacking single level affects that single level only: the system will continue applying detection to the other levels. The failure of one local intrusion detection process does not cripple an entire IDS, even though it causes some degradation of overall detection accuracy.

Finally, it improves system robustness. The IDS will be difficult to attack, as it is divided into many detection levels (depending on the number of devices in the network) that make attacking the system much more difficult. Nevertheless, there remain unresolved problems to building an effective IDS which are not covered in this thesis, such as an inability to detect new attacks and weak system reactive capabilities, etc.. More details about future work planning are given in the next section.

## 5.2 Future Research Directions

IDS modeling in this thesis has been focused on improving the detection model in terms of detection accuracy, efficiency, and scalability, without creating a heavy network load on the system. We believe there are many possible extensions for the IDS problem. Therefore, below are presented other suggestions to further improve the IDS.

- Implement the C-dIDS using autonomous agents. With autonomous agents, the architecture can be easily extended and maintained. In addition, they can enable ongoing interaction with the environment and cooperating with other agents.
- Build a testing methodology to test the proposed architecture in terms of its robustness, security, feasibility, reliability, and other criteria, and compare it with other dIDS architectures.
- Investigate the feasibility of implementing the C-dIDS in real-time intrusion detection environments.

- Security, or secure message handling between system components, by using different approaches to accomplish confidentiality, integrity, and authentication for communications, could prevent the blocking of messages or the generation of false messages.

We have already implemented the first part of the IDS, which is the detection model. We need to focus on the other IDS parts:

- Response Mechanisms: most current IDS implementations have limited response (reactive) capabilities; an IDS needs to be capable of preventing, not just reporting an attack. We are planning to extend our study to build an efficient response system.
- Build a friendly interface agent. The user interface is an important issue for future work. Most of the recent works in IDS has focused on how to implement detection, but very little has been done in the way of presenting the information to the user, or how to allow the user to specify policies that the IDS can understand and enforce. IDS should offer a user interface to facilitate better control over intrusion detection activity and better understanding of the alert information.

- Investigate other major limitations of IDS such as:
  - New and Unknown Attack Recognition: The intelligent method of attack detection will be researched further to overcome the problem of detection of unknown and novel attack forms.
  - Dynamic nature: Provided with a dynamic nature, the IDS automatically learns new intrusion methods on their own, without a central controller having predefined information. This area needs to be studied deeply through the interaction with changing network environments, various security requirements, and other intrusion detection processes.

- There are several areas where C-dIDS requires additional work before it can become more responsive to the demands of a wide range of environments prevalent in networking applications, such as:
  - A layered framework for the placement of dIDS devices needs to be investigated.
  - The effectiveness of dIDS depends also on how much of the data traffic is transferred between the system components on the distributed environment (in our case the components are the IDS modules). Therefore, the relationship between the proposed architecture and traffic needs to be explored in order to improve the overall dIDS effectiveness.

- We did not consider the balance between system performance and system cost. High performance always entails a high system cost. We will do more work on improving system performance with a reasonable system cost.

Finally, IDS are not the answer to all network security problems. They require a certain level of maturity and are only effective if monitored and maintained. This thesis is only one among many preliminary starts in the field. There are many topics for possible future work, but we hope that our work will be of service to the growing population of IDS users and researchers.

# Appendices

## Appendix A: A Description of DARPA Dataset

The DARPA KDD-99 dataset [111] is based on DARPA 98 Intrusion detection dataset, which aims to provide data for researchers working on intrusion detection in general. The DARPA KDD-99 dataset contains network data to configure and evaluate intrusion detection systems. This recorded network data contains 22 attack types and normal connections.

The attack types are:

```
(1)  back
(2)  buffer_overflow
(3)  ftp_write
(4)  guess_passwd
(5)  imap
(6)  ipsweep
(7)  land
(8)  loadmodule
(9)  multihop
(10) neptune
(11) nmap
(12) perl
(13) phf
(14) pod
(15) portsweep
(16) rootkit
(17) satan
(18) smurf
(19) spy
(20) teardrop
(21) warezclient
(22) warezmaster
```

In the KDD-99, a connection is represented by 41 features, 22 of these features describe the connection itself and 19 of them describe the properties of connections to the same host in the same host in last two seconds (IDS should analyze the service types used by the same user in previous connections and for this purpose there are 10 features included in the 41 feature vector). These 41 features can be divided into different groups:

***Basic features***:

(1) Duration of connection

(2) Protocol type (3 different symbols: TCP, UDP, ICMP)

(3) Service type (70 different symbols: FTP, HTTP, Telnet...)

(4) Status flag (11 different symbols)

(5) Total bytes sent to destination host

(6) Total bytes sent to source host

(7) Whether source and destination addresses are the same or not

(8) Number of wrong fragments

(9) Number of urgent packets

(10-41) 32 derived features, falling into three categories:

(1) ***Content features***: domain knowledge is used to assess the payload of the original TCP packets. (Ex: number of failed login attempts)

(2) ***Time-based traffic features***: these features are designed to capture properties that mature over a two seconds temporal window. (Ex: number of connections to the same host over the two seconds interval)

(3) ***Host based traffic features***: utilize a historical window estimated over the number of connections. Host-based features are therefore designed to assess attacks, which span interval longer than two seconds.

The 41 features are as the following:

```
(1)  duration: continuous.
(2)  protocol_type: symbolic.
(3)  service: symbolic.
(4)  flag: symbolic.
(5)  src_bytes: continuous.
(6)  dst_bytes: continuous.
(7)  land: symbolic.
(8)  wrong_fragment: continuous.
(9)  urgent: continuous.
(10) hot: continuous.
(11) num_failed_logins: continuous.
(12) logged_in: symbolic.
(13) num_compromised: continuous.
(14) root_shell: continuous.
(15) su_attempted: continuous.
(16) num_root: continuous.
(17) num_file_creations: continuous.
```

```
(18) num_shells: continuous.
(19) num_access_files: continuous.
(20) num_outbound_cmds: continuous.
(21) is_host_login: symbolic.
(22) is_guest_login: symbolic.
(23) count: continuous.
(24) srv_count: continuous.
(25) serror_rate: continuous.
(26) srv_serror_rate: continuous.
(27) rerror_rate: continuous.
(28) srv_rerror_rate: continuous.
(29) same_srv_rate: continuous.
(30) diff_srv_rate: continuous.
(31) srv_diff_host_rate: continuous.
(32) dst_host_count: continuous.
(33) dst_host_srv_count: continuous.
(34) dst_host_same_srv_rate: continuous.
(35) dst_host_diff_srv_rate: continuous.
(36) dst_host_same_src_port_rate: continuous.
(37) dst_host_srv_diff_host_rate: continuous.
(38) dst_host_serror_rate: continuous.
(39) dst_host_srv_serror_rate: continuous.
(40) dst_host_rerror_rate: continuous.
(41) dst_host_srv_rerror_rate: continuous.
```

Sample of DARPA dataset

0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal

0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00, portsweep

# Bibliography


[1] Y. Wu, B. Foo, Y. Mei, and S. Bagchi. "Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS", *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 03),* Page(s): 234 – 244, 8-12 December 2003.

[2] J. Yu; Reddy, R. Reddy, S. Selliah, S. Kankanahalli. S. Reddy, and V. Bharadwaj, "A Collaborative Architecture for Intrusion Detection Systems with Intelligent Agents and Knowledge-Based Alert Evaluation", *Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design*, Vol. 2, Page(s): 271 – 276, 26-28 May 2004.

[3] R. Zhang, D. Qian, H. Chen, and W. Wu, "Collaborative Intrusion Detection Based on Coordination Agent", *Proceedings of the Fourth International Conference in Parallel Distributed Computing, Applications and Technologies,* Page(s): 175- 179, 27-29 August 2003.

[4] D. Ye, W. Hui-Qiang, and P. Yong-Gang, "Design of a Distributed Intrusion Detection System Based on Independent Agents", *Proceedings of the International Conference on Intelligent Sensing and Information,* Page(s): 254 – 257, 2004.

[5] Q. Xue, L. Guo, and J. Sun, "The Design of a Distributed Network Intrusion Detection System IA-NIDS". *Proceedings of the International Conference on Machine Learning and Cybernetics*, Vol.4, Page(s): 2305- 2308, 2-5 November 2003.

[6] F. Karray, and C. Silva, "Soft Computing and Intelligent Systems Design: Theory, Tools and Applications", Addison Wesley Publishing, 2004.

[7] J. Hertz, A. Krogh, and R. Palmer, "Introduction to the Theory of Neural Computation", Addison Wesley Publishing, 1991.

[8] L. Silva, A. Santos, J. Silva, and A. Montes, "A Neural Network Application for Attack Detection in Computer Networks", *Proceedings of the IEEE International Joint Conference on Neural Network,* Vol.2, Page(s):1569 – 1574, 25-29 July 2004.

[9] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts. "Network-Based Intrusion Detection Using Neural Networks", *unpublished technical report, Rensselaer Polytechnic Institute, Troy, New York.* http://www.cs.rpi.edu/~szymansk/papers/annie02.pdf (March 2009).

[10] S. Mukkamala and A. Sung, "Artificial Intelligent Techniques for Intrusion Detection", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,* Vol.2, Page(s): 1266- 1271, October 2003.

[11] P. Tillapart, Th. Thumthawatworn and P. Santiprabhob, "Fuzzy Intrusion Detection System", Vol.6, No. 2, Page(s): 109-114, October 2002.

[12] Z. Jian, D. Yong, and G. Jian, "Intrusion Detection System Based on Fuzzy Default Logic", *Proceedings of the the 12th IEEE International Conference on Fuzzy Systems,* Vol.2, Page(s): 1350- 1356, May 2003.

[13] M. Pillai, j. Eloff, and H. Venter, "An Approach to Implement a Network Intrusion Detection System using Genetic Algorithms", *Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries,* Vol. 75, Page(s): 221 – 221, 2004.

[14] A. Chittur, "Model Generation for an Intrusion Detection System Using Genetic Algorithms", November 27, 2001.

[15] W. Li, "Using Genetic Algorithm for Network Intrusion Detection", Unpublished technical report. Department of Computer Science and Engineering, Mississippi State University. http://www.security.cse.msstate.edu/docs/Publications/wli/DOECSG2004.pdf.

[16] C. Zhang, J. Jiang, and M. Kamel, "Intrusion Detection using Hierarchical Neural Networks", *Pattern Recognition Letters 26*, Page(s): 779–791, 16 February 2004.

[17] J. Mill and A. Tnoue, "Support Vector Classifiers and Network Intrusion Detection", *Proceedings of the IEEE International Conference on Fuzzy Systems,* Page(s): 407- 410, 25-29 July 2004.

[18] K. Li, H. Huang, S. Tian and W. Xu, "Improving One-Class SVM for Anomaly Detection", *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, Page(s): 3077- 3081, 2-5 November 2003.

[19] A. Sung and Srinivas Mukkamala, "Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks", *Symposium on Application and Internet (SAINT'03),* Page(s): 209- 216, 27-31 January. 2003.

[20] V. Golovko, L. Vaitsekhovich, P. Kochurko and U. Rubanau, "Dimensionality Reduction and Attack Recognition using Neural Network Approaches", *Proceedings of the International Joint Conference on Neural Networks,* Page(s): 2734-2739, 12-17 August 2007.

[21] K. Shazzad and J. Park, "Optimization of Intrusion Detection through Fast Hybrid Feature Selection", *Proceeding of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2005, (PDCAT'05),* Page(s): 264 – 267, 05-08 December 2005.

[22] A. Hofmann, T. Horeis, and B. Sick, "Feature Selection for Intrusion Detection: An Evolutionary Wrapper Approach", *Proceedings of the IEEE International Joint Conference on Neural Networks,* Page(s): 1563- 1568, 25-29 July 2004.

[23] D. Kim, H. Nguyen, and J. Park, "Genetic Algorithm to Improve SVM Based Network Intrusion Detection System", *Proceedings of the 19th International Conference on Advanced Information Networking and Applications, 2005 (AINA 2005),* Page(s): 155 – 158, 28-30 March 2005.

[24] A. Hofmann and B. Sick, „Evolutionary Optimization of Radial Basis Function Networks for Intrusion Detection", *Proceedings of the international Joint Conference on Neural Networks,* Page(s): 415- 420, 20-24 July 2003.

[25] S. Mukkamala and A. Sung, "Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines", *Technical Report*, Unpublished paper.

[26] Mukkamala and A.Sung, "Detecting Denial of Service Attacks Using Support Vector Machines", *Proceedings of the 12th IEEE International Conference on Fuzzy Systems, 2003,* Page(s): 1231 – 1236, 25-28 May 2003.

[27] H. Gao, H. Yang, and X. Wang, "Ant Colony Optimization Based Network Intrusion Feature Selection and Detection", *Proceedings of The Fourth International Conference on Machine Learning and Cybernetics,* Page(s): 18-21, August 2005.

[28] S. Srinoy, "Intrusion Detection Model Based On Particle Swarm Optimization and Support Vector Machine", *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2007)*, Page(s): 186-192, 1-5 April 2007.

[29] P. Lichodzijewski, A. Zincir-Heywood, and M. Heywood, "Dynamic Intrusion Detection Using Self-Organizing Maps", *Unpublished technical report*, Dalhosie University, Halifax. http://citeseer.ist.psu.edu/lichodzijewski02dynamic.html. (March 2009)

[30] W. Jing-xin, W. Zhi-ying, and D. Kui, "A Network Intrusion Detection System based on the Artificial Neural Networks:, *Proceedings of the 3rd international conference on Information security, ACM International Conference Proceeding Series*, Vol. 85, Page(s): 166 – 170, 2004.

[31] J. Lei and A. Ghorbani, "Network Intrusion Detection Using an Improved Competitive Learning Neural Network", *Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04), IEEE Computer Society*, Page(s): 190 – 197, 2004.

[32] P. Lichodzijewski and A. Zincir, "Host-Based Detection Using Self-Organizing Maps", *Proceedings of the 2002 International Joint Conference on Neural Networks,* Vol. 2, Page(s): 1714-1719, 2002.

[33] J. Li, G. Zhang, and G. Gu, "The Research and Implementation of Intelligent Intrusion Detection System Based on Artificial Neural Network", *Proceedings of the Third IEEE International Conference on Machine Laming and Cybernetics, Shanghai,* Vol. 5, Page(s):3178 – 3182, August 2004.

[34] S. Mukkamala and A. Sung, "A Framework for Countering Denial of Service Attacks", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,* Page(s): 3273 – 3278, 10-13 October 2004.

[35] Y. Chen, A. Abraham, B. Yang, "Hybrid Flexible Neural-Tree-Based Intrusion Detection Systems",. *International Journal of Intelligent Systems*. Vol. 22, No. 2, Page(s): 337 – 352, 2007.

[36] X. Li and N. Ye, "Mining Normal and Intrusive Activity Patterns for Computer Intrusion Detection", *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*. Page(s): 226-238, August 2004.

[37] J. Dickerson, J. Juslin, O. Koukousoula, and J. Dickerson, "Fuzzy Intrusion Detection", *Proceedings of the International Joint 9th Conference on IFSA World Congress and 20th NAFIPS*, Vol. 3, Page(s): 1506-1510, 25-28 July 2001.

[38] S. Zanero and S. Savaresi, "Unsupervised Learning Algorithms for Intrusion Detection", *Proceedings of the 2004 ACM Symposium on Applied Computing*, Pages: 412 – 419, 2004.

[39] Q. Wang and V. Megalooikonomou, "A Clustering Algorithm for Intrusion Detection", *Proceedings of the SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, Vol. 5812, Page(s): 31-38, March 28 - April 1, 2005.

[40] D. Novikov, R. Yampolskiy, and L. Reznik, "Artificial Intelligence Approaches for Intrusion Detection", *Proceedings of the IEEE Conference on Systems, Applications and Technology,* Page(s): 1-8, May 2006.

[41] S. Snapp, J. Brentano, G. Dias, T. Goan, T. Grance, L. Heberlein, C. Ho, K. Levitt, B. Mukherjee, D. Mansur, K. Pon, and S. Smaha, "A System for Distributed Intrusion Detection [C]", *Proceedings of the 14th Conference on National Computer Security Conference*, Vol.9, Page(s): 170-176, 25 Feb-1 March 1991.

[42] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS: A Graph Based Intrusion Detection System for Large Networks", In *Proceedings of the 19th National Information Systems Security Conference*, Vol. 1, Page(s): 361–370, October 1996.

[43] P. Porras and P. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances", *Proceedings of the 20th National Information Systems Security Conference*, 1997.

[44] E. Spafford and D. Zamboni, "Intrusion Detection using Autonomous Agents", *The International Journal of Computer and Telecommunications Networking*, Page(s): 547-570, 2000.

[45] J. Hochberg, K. Jackson, C. Stallings, J. McClary, D. DuBois, and J. Ford, "NADIR: An Automated System for Detecting Network Intrusion and Misuse", *Proceedings of the Conference on Computers and Security*, Page(s): 235–248, May 1993.

[46] G. White, E. Fisch, and U. Pooch, "Cooperating Security Managers: A Peer-Based Intrusion Detection System", *IEEE Net-work*, Vol. 10, No. 1, Pags(s): 20–23, January/February 1996.

[47] M. Slagell, "The Design and Implementation of MAIDS (Mobile Agent Intrusion Detection)", *Technical Report TROI-07, Iowa State University, Department of Computer Science,* 2001.

[48] J. Li and C. Manikopoulos, "Early Statistical Anomaly Intrusion Detection of DOS Attacks using MIB Traffic Parameters", *Information Assurance Workshop, IEEE Systems, Man and Cybernetics Society.* Page(s): 53 – 59, 18-20 June 2003.

[49] W. Teng, M. Hsieh and M. Chen, "A statistical framework for mining substitution rules", *Knowledge and Information Systems Journal*, Vol. 7, No. 2, Page(s): 158 – 178, 2005.

[50] Th. Verwoerd and R. Hunt, "Intrusion Detection Techniques and Approaches", *Journal in Computer Communications*, Page(s): 1356-1365. 2002.

[51] W. Lee and D. Xiang, "Information-Theoretic Measures for Anomaly Detection", *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, Page: 130, 2001.

[52] P. Williams, K. Anchor, J. Bebo, G. Gunsch, and G. Lamont, " CDIS: Towards a Computer Immune System for Detecting Network Intrusions", *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection 2001*, Page(s): 117-133, 2001.

[53] Ch. Ehret and U. Ultes-Nitsche, „Immune System Based Intrusion Detection System", Technical Report, http://icsa.cs.up.ac.za/issa/2008/Proceedings/Full/50.pdf. (March 2009)

[54] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology", *Book: Communications of the ACM,* Vol. 40, No. 10, Page(s): 88-96, 1997.

[55] S. Microsystems, " Installing, Administering, and Using the Basic Security Module", December 1991.

[56] V.Paxson, "Bro: A System for Detecting Network Intruders in Real-Time. *Proceedings of the Symposium on USENIX Security*, January 1998, ftp://ftp.ee.IbI.gov/papers/bro-usenix98-revised.ps.Z. (March 2009)

[57] Snort-the Open Source Network Intrusion Detection System, http://www.snort.org/. (March 2009)

[58] S. Kumar and E. Spafford, "Pattern Matching Model for Misuse Intrusion Detection. *Proceedings of the 17th National Computer Security Conference*, 1994.

[59] Z. Chunyue, L. Yun, and Z. Hongke, "A Pattern matching based Network Intrusion Detection System", *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV '06),* Page(s): 1-4, December 2006.

[60] K. Ilgun, R. Kemmerer, and P. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection Approach", *IEEE Transactions on Software Engineering*, Vol. 20, No. 5, March 1995.

[61] A. Mounji, "Languages and Tools for Rule-Based Distributed Intrusion Detection", *PhD thesis, Facult´es Universitaires Notre-Dame de la Paix Namur (Belgium)*, September 1997.

[62] syslog(3). UNIX documentation.

[63] Common Intrusion Detection Framework Working Group, "A CISL" Tutorial. http://www.gidos.org/tutorial.html. (March 2009)

[64] D. Curry, "Intrusion Detection Message Exchange Format: Extensible Markup Language (XML) Document Type Definition", January 2009.

[65] S. Eckmann, G. Vigna, and R. Kemmerer, "STATL", Technical report, UCSB, 2000.

[66] S. Eckmann, G. Vigna, and R. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection". *Journal of Computer Security*, Vol. 10, No. 1/2, Page(s): 71-104, 2002.

[67] J. Balasubramaniyan, J. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents" *Proceedings of the Fourteenth Annual Computer Security Applications Conference*, Page(s): 13–24, December 1998.

[68] U. Lindqvist and P.A. Porras, "Detecting Computer and Network Misuse with the Production-Based Expert System Toolset (P-BEST)", *Proceedings of the IEEE Symposium on Security and Privacy*, May 1999.

[69] Secure Networks, "Custom Attack Simulation Language (CASL)", January 1998.

[70] R. Deraison, "The Nessus Attack Scripting Language Reference Guide", 2000. http://www.nessus.org. (March 2009)

[71] M. Ranum, K. Landfield, M. Stolarchuck, M. Sienkiewicz, A. Lambeth, and E. Wall, "Implementing a Generalized Tool for Network Monitoring", *Proceedings of the Eleventh Conference on Systems Administration (LISA '97), USENIX,* October 1997.

[72] C. Ko, M. Ruschitzka, and K. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach", *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, Page(s): 175–187, 1997.

[73] R. Sekar and P. Uppuluri, "Synthesizing Fast Intrusion Detection/Prevention Systems from High-Level Specifications", *Proceedings of the USENIX Security Symposium*, 1999.

[74] W. Lee, S. Stolfo, Ph. Chan, E. Eskin, W. Fan, M. Miller, Sh. Hershkop and J. Zhang, "Real Time Data Mining-based Intrusion Detection", *Proceedings of DISCEX II*, June 2001.

[75] S Terry Brugger, "Data Mining Methods for Network Intrusion Detection", UC Davis Dissertation Proposal, 9 June 2004.

[76] D. Dasgupta, "Immunity-based Intrusion Detection Systems: A General Framework", *Proceedings of the 22nd National Information Systems Security Conference*, Page(s): 18-21, 1999.

[77] R. Robbins, "Distributed Intrusion Detection Systems: An Introduction and Review", *GSEC Practical Assignment*, version 1.4b, Option 1, January 2, 2002.

[78] G. Kim and E. Spafford, "Experience with Tripwire: Using Integrity Checkers for Intrusion Detection", *Systems Administration, Networking and Security Conference III*, USENIX, 1994.

[79] M. Ranum, "Artificial Ignorance: How-to guide. Firewall Wizards Mailing List", http://lists.insecure.org/firewall-wizards/1997/Sep/0096.html. (March 2009)

[80] E. Eiland and L. Liebrock, "An Application of Information Theory to Intrusion Detection", *Proceedings of the Fourth IEEE International Workshop on Information Assurance*, Page(s): 119 – 134, 2006.

[81] M. Crosbie, B. Dole, T. Ellis, I. Krsul, and E. Spafford, "IDIOT Users Guide", Purdue University, 1996.

[82] J. Xin, J. Dickerson, and J. Dickerson, "Fuzzy Feature Extraction and Visualization for Intrusion Detection", *Proceeding of the 12th IEEE International Conference on Fuzzy Systems*, Vol.2, Page(s): 1249- 1254, 25-28 May 2003.

[83] J. Tian, Y. Fu, Y. Xu, and J. Wang, "Intrusion Detection Combining Multiple Decision Trees by Fuzzy logic", *The Sixth International Conference on Parallel and Distributed Computing, Application and Technologies PDCAT 2005*, Page(s): 256- 258, 05-08 December 2005.

[84] Y. Dhanalakshmi and I. Babu, "Intrusion Detection Using Data Mining Along Fuzzy Logic and Genetic Algorithms", *IJCSNS International Journal of Computer Science and Network Security*, Vol.8, No.2, February 2008.

[85] N. Bashah, I. Shanmugam, and A. Ahmed, "Hybrid Intelligent Intrusion Detection System", *Proceedings of World Academy of Science, Engineering and Technology*, Vol. 6, Page(s): 1307-6884, June 2005.

[86] J. Gomez and D. Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection", *Proceedings of the 2002 IEEE Workshop on Information Assurance*, June 2001.

[87] P. Cunningham. Dimension Reduction, Technical Report. UCD-CSI-2007-7. August 2007.

[88] I. Fodor, "A Survey of Dimension Reduction Techniques", Technical Report UCRL-ID-148494, Lawrence Livermore Nat'l Laboratory, Center for Applied Scientific Computing, June 2002.

[89] M. Treaster, "A Survey of Distributed Intrusion Detection Approaches", ArXiv Computer Science e-prints: cs/0501001. December 2005, Available at: http://arxiv.org/abs/cs/0501001. (March 2009)

[90] G. John , R. Kohavi, and K. Pleger, "Irrelevant Features and the Subset Selection Problem", *Proceeding of the 11th Int. Conference on Machine Learning*, Page(s): 121-129, 1994.

[91] S. Raudys and A. Jain, "Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners*", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 3, Page(s): 252-264, March 1991.

[92] A. Jain, R. Duin, and J. Mao, "Statistical Pattern Recognition: A Review", *IEEE Transaction* on *Pattern Analysis and Machine Intelligence,* Vol. 22, No. 1, Page(s): 4-37, January 2000.

[93] A.R. Webb, "Statistical Pattern Recognition", second edition, Wiley, 2002.

[94] M. Ringner, "What is Principal Component Analysis?", Nature Publishing Group, Vol. 26, No. 3, Page(s): 303-304, March 2008. http://www.nature.com/naturebiotechnology

[95] K. Teknomo, "Discriminant Analysis" Tutorial. http://people.revoledu.com.proxy.lib.uwaterloo.ca/kardi/ tutorial/LDA/ (March 2009)

[96] H. Peng, F. Long, and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 8, Page(s):1226 – 1238, August 2005.

[97] P. Mitra, C. Murthy, and S. Pal, "Unsupervised Feature Selection Using Feature Similarity", *IEEE Transaction on Pattern Analysis and Machine Intelligence,* Vol. 24, No. 3, Page(s): 301-312, March 2002.

[98] S. Snapp, J. Brentano, and G. Dias, "DIDS (Distributed Intrusion Detection System) – motivation, architecture, and an early prototype", *Proceedings of the 14th National Computer Security Conference*, October 1991.

[99] J. Barrus and N. Rowe, "A Distributed Autonomous-Agent Network-Intrusion Detection and Response System", *Proceedings of the Symposium on Command and Control Research and Technology*, Page(s): 577-586, June 1998.

[100] B. Krishnapuram, A. Hartemink, L. Carin, and M. Figueiredo, "A Bayesian Approach to Joint Feature Selection and Classifier Design", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 9, Page(s): 1105-1111, September 2004.

[101] M. Law, M. Figueiredo, and A. Jain, "Simultaneous Feature Selection and Clustering Using Mixture Models", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 9, Page(s): 1154-1166, September 2004.

[102] S. Pal, R. De, and J. Basak. "Unsupervised Feature Evaluation: A Neuro-Fuzzy Approach." *IEEE Transaction on Neural Networks*, Vol. 11, No. 2, Page(s): 366-376, March 2000.

[103] Hua-Liang Wei and Stephen A. Billings, "Feature Subset Selection and Ranking for Data Dimensionality Reduction", *IEEE Transactions on Pattern Analysis And Machine Intelligence*, Vol. 29, No. 1, Page(s):162 – 166, January 2007.

[104] H. Liu, L. Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering", *IEEE Transaction on Knowledge and Data Engineering*, Vol. 17, No.: 4, Page(s): 491–502, April 2005.

[105] A. Tamilarasan, S. Mukkamala, A. Sung, and K. Yendrapalli, "Feature Ranking and Selection for Intrusion Detection Using Artificial Neural Networks and Statistical Methods", *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN'06),* Page(s):4754 - 4761, July 16-21, 2006.

[106] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision Tree Classifier For Network Intrusion Detection With GA-based Feature Selection", *Proceedings of the 43rd ACM Southeast Conference*, March 2005.

[107] M. Dash, H. Liu, and H. Motoda, "Consistency based feature selection", *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2002),* Page(s): 98-109, 2002.

[108] H. Almuallim and T. Dietterich, "Learning Boolean Concepts in the Presence of Many Irrelevant Features", Artificial Intelligence, Vol. 69, No. 1-2, Page(s): 279-305, 1994.

[109] K. Kira and L. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm", *Proceedings of the AAAI-92*, Page(s): 129-134, 1992.

[110] H. Almuallim and T. G. Dietterich, " Learning with Many Irrelevant Features", *Proceedings of the AAAI-91*, Page(s): 547-551, 1991.

[111] http://www.ll.mit.edu/mission/communications/ist/index.html (March 2009)

[112] http://archive.ics.uci.edu/ml/ (March 2009)

[113] G. Loosli. Toolbox SimpleSVM Documentation. http://cbio.ensmp.fr/sirene/documentationSimpleSVM.pdf (March 2009)

[114] I. Guyon, J. Weston, S. Barnhill, M.D. and V. Vapnik, "Gene Selection for Cancer Classification using Support Vector Machines", *Journal on Machine Learning*, Page(s): 389-422, October 31, 2004.

[115] I. Onut and A. Ghorbani, "A Feature Classification Scheme for Network Intrusion Detection", *International Journal of Network Security*, Page(s): 1–15, July 2007.

[116] I. Onut and A. Ghorbani, "Features vs. Attacks: A Comprehensive Feature Selection Model for Network Based Intrusion Detection Systems", Lecture notes in Computer Science, Page(s): 19–36, Springer-Verlag Berlin Heidelberg 2007.

[117] D. Novikov, R. Yampolskiy, and L Reznik, "Anomaly Detection Based Intrusion Detection", *Proceedings of the third International Conference on Information Technology,,* Page(s):420 – 425, April 2006.

[118] S. Zaman S., F. Karray, „Fuzzy ESVDF approach for Intrusion Detection System" *Proceedings of the 23rd IEEE International Conference on Advanced Information Networking and Applications (AINA-09),* May 26-29, 2009.

[119] S. Zaman and F. Karray, "Features Selection Using Fuzzy ESVDF for Data Dimensionality Reduction", *Proceedings of the International Conference on Computer Engineering and Technology ICCET'08*, Vol. 1, Page(s): 81-87, 22-24 January 2009.

[120] F. Barika, N. Kadhi, K. Ghedira, "Intelligent and Mobile Agent for Intrusion Detection System: IMA-IDS", Technical Report, November 2003.

[121] D. Ye, W. Hui-Qiang, and P. Yong-Gang, "Design of A Distributed Intrusion Detection System Based on Independent Agents", *Proceedings of International Conference on Intelligent Sensing and Information Processing,* Page(s): 254 – 257, 2004.

[122] B. Mukherjee, L.T. Heberlein, and K.N. Levitt, "Network Intrusion Detection" *IEEE Network*, Page(s): 26-41, Vo1.8, No.3, May-June 1994.

[123] M. Yasin and A. Awan, "A Study of Host-Based IDS using System Calls", *Proceedings of the International Conference on Networking and Communication 2004*, Page(s): 36- 41, June 2004.

[124] R. Heady, G. Luger, A. Maccabe, and M. Servilla, "The architecture of a network level intrusion detection system", Technical Report CS90-20, Department of Computer Science, University of New Mexico, August 1990.

[125] S. Lee and D. Heinbuch, " Training a Neural-Network Based Intrusion Detector to Recognize Novel Attacks", *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems And Humans*, Vol. 31, No. 4, Page(s): 294-299, July 2001.

[126] N. Ye, X. Li, Q. Chen, S. Emran, and M. Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data". *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems And Humans*, Vol. 31, No. 4, July 2001.

[127] W. Hu, W. Hu, S. Maybank, "AdaBoost-Based Algorithm for Network Intrusion Detection", *IEEE Transactions on Systems, Man, and Cybernetics —Part B*, Vol. 38, No. 2, Page(s): 577-583, April 2008.

[128] D. Dasgupta and F. Gonzalez, "An Immunity-based Technique to Characterize Intrusions in Computer Networks". *IEEE Transactions on Evolutionary Computing*, Vol. 6, No. 3, Page(s): 281-291, June 2002.

[129] D. Russell and G. Gangemi, "Computer Security Basics", O'Reilly & Associates, Sebastopol, CA, 1991.

[130] S. Kumar, "Classification and Detection of Computer Intrusions", PhD thesis, Purdue University, 1995.

[131] J. Anderson, "Computer security threat monitoring and surveillance", Technical Report, James P. Anderson Co., Fort Washington, PeNNylvania, 1980,

[132] D. Denning. "An intrusion detection model". *IEEE Transactions on Software Engineering,* Vol. 13, No.: 2, Page(s): 222–232, 1987.

[133] S. Smaha, "Haystack: An intrusion detection system", *Proceedings of the 14th Conference on Aerospace Computer Security Applications*, Page(s): 37–44, 1988.

[134] L. Heberlein. K. Levitt, and B. Mukherjee, "A method to detect intrusive activity in a networked environment", *Proceedings of the 14th Conference on National Computer Security Conference*, Page(s): 362–371, 1991.

[135] T. Heberlein and M. Bishop, "Attack Class: Address Spoofing", Addison- Wesley Pub Co, 1998.

[136] A. Rapaka, A. Novokhodko, and D. Wunsch, "Intrusion Detection Using Radial Basis Function Network on Sequences of System Calls", *Proceedings of the International Joint Conference on Neural Networks,* Page(s): 1820- 1825 Vol.3, July 2003.

[137] M. Moradi and M. Zulkernine, "A Neural Network Based System for Intrusion Detection and Classification of Attacks", Unpublished technical report, this work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). http://www.cs.queensu.ca/~moradi/148-04-MM-MZ.pdf.

[138] S. Mukkamala and A. Sung, "A Comparative Study of Techniques for Intrusion Detection", *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence,* Page: 570, 2003.

[139] H. Debar, "An Introduction to Intrusion-Detection Systems", *Proceedings of Connect'2000*, April 29th-May 1st, 2000.

[140] P. Williams, K. Anchor, J. Bebo, G. Gunsch, and G. Lamont, "CDIS: Towards a Computer Immune System for Detecting Network Intrusions", *Proceedings 4th Int'l Symposium, Recent Advances in Intrusion Detection*, Page(s): 117–133, 2001.

[141] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus, *"Knowledge Discovery Database: an Overview",* AI Magazine, Page(s):57-70 Vol. 13, 1992.

[142] J. Pei, Sh. Upadhyaya, F. Farooq, and V. Govindaraju, "Data Mining for Intrusion Detection: Techniques, Applications and Systems", *Proceedings 20th International Conference on Data Engineering,* Page(s): 877- 877, 30 March-2 April 2004.

[143] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang and S. Zhou, "Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions", *Proceedings of the 9th ACM conference on Computer and communications security,* Page(s): 265 – 274, 2002.

[144] P. Uppuluri and R. Sekar, "Experiences with Specification-based Intrusion Detection", *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, Lecture Notes In Computer Science,* Vol. 2212, Page(s): 172 – 189, 2001.

[145] S. Webster, "The Development and Analysis of Intrusion Detection Algorithms", *M.S. Thesis, Massachusetts Institute of Technology*, 1998.

[146] H. Kai, H. Zhu, K. Eguchi, N. Sun, and T. Tabata, "A Novel Intelligent Intrusion Detection, Decision, Response System*", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences archive*, Vol. E89-A, Page(s): 1630-1637, June 2006.

[147] A. Curtis, and J. Carver, "Intrusion Response Systems: A Survey", Technical Report, Department of Computer Science, Texas A&M University, 2000.

[148] R. Sielken and A. Jones, "Application Intrusion Detection Systems: The Next Step". *ACM Transactions on Information Security,* August 1999.

[149] R. Sielken, "Application Intrusion Detection Source", Technical Report: CS-99-17, 1999.

[150] B. Thomas, "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms", 1996.

[151] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", *Master's Thesis, Massachusetts Institute of Technology*, 1998.

[152] S.Zaman and F. Karray, "TCP/IP Model and Intrusion Detection Systems", *Proceedings of the IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA-09),* 26-29 May 2009.

[153] S. Zaman, and F. Karray, "Collaborative Architecture for Distributed Intrusion Detection System based on Lightweight IDS Modules", *Proceedings of the 2009 IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT),* 29-30 August 2009.

[154] S. Zaman, and F. Karray, "A Collaborative Architecture for Distributed Intrusion Detection System", *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defense Applications,* 8-20 July 2009.

[155] A. Abraham, R. Jain, J. Thomas, S. Han, "D-SCIDS: Distributed Soft Computing Intrusion Detection System", *Journal of Network and Computer Applications*, Vol.30 , No. 1, January 2007.

[156] S. Staniford-Chen, S. Tung, D. Schnackenberg, "The Common Intrusion Detection Framework (CIDF)", *Proceedings of the information survivability workshop*, October 1998.